Exploiting cloud utility models for profit and ruin

by

Joseph Robert Idziorek

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:

Doug Jacobson, Major Professor

Tom Daniels

Yong Guan

Mani Mina

Leigh Tesfasion

Iowa State University

Ames, Iowa

2012

DEDICATION

I would like to dedicate this dissertation to Arlowyn, the love of my life, and to my family and friends for their kindness and constant support throughout this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

Having nearly completed my PhD program, I now know why people are so grateful in the acknowledgements section of their dissertations - the work of a dissertation is completed with the love, support, help and guidance of numerous people. If it is true that it takes a "village" to raise a child, then it is my opinion that it takes $n\cdot$villages for an individual to complete a PhD program where, for me, n is the cardinality of a set A which includes, but is not limited to: $A = \{family, friends, ISU, SCSU, HHS, IBM\}$. I would like to take this opportunity to express my sincere thanks and gratitude to those who have helped me along the way through a very fulfilling, turbulent, enjoyable, and taxing journey known as grad school. Although a bit long winded, there are very few times in life where one can publicly acknowledge those around them and I am going to make the best of this opportunity.

Dr. Jacobson, thank you for taking a chance on me as a graduate student. Your unwavering support and the confidence you bestowed upon me relieved more anxiety than you will ever know. Thank you for enabling me to explore the research topics that were of interest to me, for allowing me to teach CprE/InfAs 131, and the opportunity to co-author a textbook to name a few. Your hard work and your success enabled me to grow as a researcher, a writer and a teacher more than I ever could have dreamed.

Mom and Dad, thank you for working so hard your entire lives to provide me with the foundation and opportunities necessary to get to this point in my life. Your constant love, encouragement and guidance has been vital throughout all my years in school. Thank you for all your sacrifices, selflessness, and for always investing in my future.

Arlowyn, my fiance, you have and will always be my motivation. Thank you for sacrificing and enduring through the two years we spent apart. I will always treasure our time in Ames together and cannot wait to continue the next chapter in the adventures of Arlowyn & Joe.

Katie, without your unjustified confidence in me, which will always be my opinion, I never

would have completed grad school. When times were the hardest and the outlook looked bleak, you were always there on the phone to keep me positive and to encourage me to continue this journey. Thank you for all of your support, encouragement, help, and care packages; I will forever be in your debt.

Mark, m-money, I never could have done this without you. We did a lot of laughing, even more studying, writing and editing, and a fair share of spirited debate. I feel very thankful to have gone through grad school with you and will always be thankful for your contributions.

Justin, Jesse, and Josh, thank you for all the weekend trips to Ames. Those visits were a lot of fun and were greatly appreciated by a very lonely grad student.

Andy, Josh and Jusitn, your friendship and camaraderie pushed me to achieve goals I never thought possible. Without meeting you gentlemen as an undergrad, I never would have made it through D-ham's or AAA's classes or senior design and, needless to say, I would never have had to confidence to apply to grad school.

Ginny, I am so thankful for your kindness, friendship, and willingness to help me navigate through the myriad of obstacles necessary to graduate.

John Carr, you are a good friend. As grad students we went through a lot of highs and lows. Some of my fondest memories of grad school are the four weekends that you, Arlowyn, and I spent grilling on your balcony, all the basketball and football games, and ring shopping.

Dr. Tom Daniels, Dr. Mani Mina, Dr. Yong Guan, and Dr. Leigh Tesfastion, my committee members, thank you for your guidance, time, and contributions to this work.

To many more, Chuck, Andrea, Renee, Dr. Petzold, Alex, Betty, Wayne, Vicky, Mr. Pothast, Sasha, Jane, Bethany, John P., Nathaniel, Sam Ellis, Paul, Tony, Eric, Tracy, thank you.

ABSTRACT

A key characteristic that has led to the early adoption of public cloud computing is the utility pricing model that governs the cost of compute resources consumed. Similar to public utilities like gas and electricity, cloud consumers only pay for the resources they consume and only for the time they are utilized. As a result and pursuant to a Cloud Service Provider's (CSP) Terms of Agreement, cloud consumers are responsible for all computational costs incurred within and in support of their rented computing environments whether these resources were consumed in good faith or not. While initial threat modeling and security research on the public cloud model has primarily focused on the confidentiality and integrity of data transferred, processed, and stored in the cloud, little attention has been paid to the external threat sources that have the capability to affect the financial viability of cloud-hosted services.

Bounded by a utility pricing model, Internet-facing web resources hosted in the cloud are vulnerable to Fraudulent Resource Consumption (FRC) attacks. Unlike an application-layer DDoS attack that consumes resources with the goal of disrupting short-term availability, a FRC attack is a considerably more subtle attack that instead targets the utility model over an extended time period. By fraudulently consuming web resources in sufficient volume (i.e. data transferred out of the cloud), an attacker is able to inflict significant fraudulent charges to the victim. This work introduces and thoroughly describes the FRC attack and discusses why current application-layer DDoS mitigation schemes are not applicable to a more subtle attack. The work goes on to propose three detection metrics that together form the criteria for detecting a FRC attack from that of normal web activity and an attribution methodology capable of accurately identifying FRC attack clients. Experimental results based on plausible and challenging attack scenarios show that an attacker, without knowledge of the training web log, has a difficult time mimicking the self-similar and consistent request semantics of normal web activity necessary to carryout a successful FRC attack.

CHAPTER 1. OVERVIEW

Chapter contains modified content from the following submitted journal paper:

Idziorek, J., Tannian, M. and Jacobson, D. Insecurity of Cloud Utility Models. *IEEE IT Professional*, © IEEE 2012.

1.1 Dissertation Organization

The chapters provided in this dissertation provide a logical progression of work starting with the broad analysis of the security aspects of the cloud computing model followed by a systematic dissection of the specific problem this dissertation seeks to address. Each of the respective chapters were written as individual papers. Because of this, the reader will find minor overlap in the introductory content between chapters and such overlap was not removed to enable each chapter to stand on their own as individual works. Together the combination of these chapters provide a comprehensive analysis of the Fraudulent Resource Consumption (FRC) attack on public cloud computing utility models.

Chapter 1 provides a high-level overview and introduction to the research problem addressed in this dissertation. Chapter 2 presents a broad security analysis of the public cloud computing model using the Parkerian Hexad as a system of analysis. From this survey of literature and cloud related security concerns, a specific attack on the cloud computing model referred to as the FRC attack is introduced in Chapter 3 and initial detection and attribution methodologies are considered. To better understand how an attacker would carry-out a FRC attack, Chapter 4 provides a simulation model and algorithm for generating web traffic in order to mimic

a normal client base and thus providing a worst-case attack scenario. Chapter 5 presents a FRC detection methodology capable of differentiating increases in aggregate traffic produced by legitimate clients from that of a FRC attack. Following suit, Chapter 6 provides an attribution methodology that distinguishes legitimate clients from FRC attack clients. To provide a complete analysis of the FRC attack, Chapter 7 discusses potential prevention and mitigation solutions. Lastly, Chapter 8 summarizes this work and identifies the contributions this work makes to the respective fields of study.

1.2 Introduction - Insecurity of Cloud Utility Models

Computing services that were traditionally hosted on organizations private servers and networks are being outsourced to third-party Cloud Service Providers (CSPs). Initial threat modeling on CSPs has concentrated on both the confidentiality (keeping data secret) and integrity (making sure the data has not changed) of data hosted in the public cloud. While these threats present real concerns, missing from threat models is the consideration of external threat sources that can affect the availability of Internet-facing cloud services. Availability in this context is not solely restricted to system downtime as a result of a Distributed Denial of Service (DDoS) attack, but also the long-term financial viability of being able to host services in the cloud.

A key feature that has led to the early adoption of public cloud computing is the utility pricing model that governs the cost of computing resources consumed [77]. Similar to public utilities like gas and electricity, cloud consumers only pay for the resources they consume (i.e., storage, bandwidth, and computer hours) and only for the time they are utilized. As a result and as obligated by a CSP's Terms of Agreement, cloud consumers are responsible for all computational costs incurred in their leased compute environments whether these resources were consumed in good faith or not.

Common use cases for corporations that have adopted public cloud computing include website and web-application hosting and e-commerce. Like any Internet-facing presence, these cloud-based services are equally vulnerable to Distributed Denial of Service (DDoS) attacks. Moreover, given the addition of pay-as-you-go-pricing, cloud-hosted web services are also vul-

nerable to attacks that seek to exploit the utility pricing model. While DDoS attacks are well known and the associated risks are well researched, this article will explore a comparatively more subtle attack on web-based services hosted in the cloud. The threat-source considered is an attacker (e.g. botnet) that seeks to perform a Fraudulent Resource Consumption (FRC) attack by consuming the metered bandwidth of web-based services that in-turn incurs a financial burden on the cloud consumer.

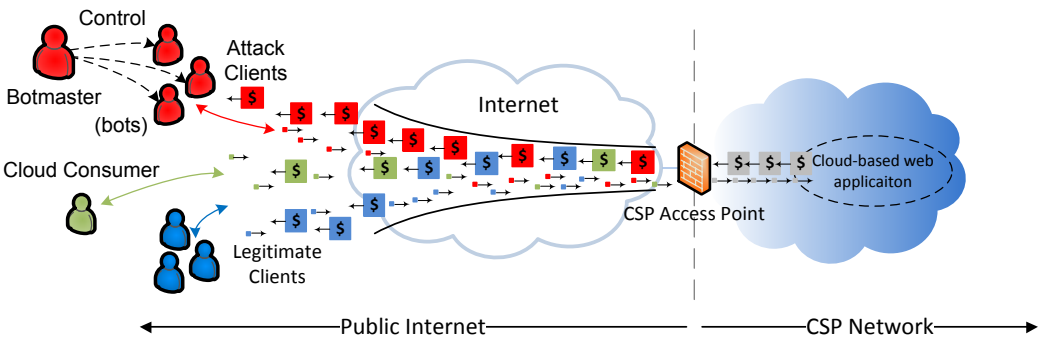


Figure 1.1: FRC Network Attack Diagram

The attack scenario depicted in Figure 1.1 illustrates the exploitation of the cloud utility model vulnerability. Here a botnet consisting of potentially thousands of bot clients is consuming web resources hosted in the cloud by mimicking legitimate client behavior. To the cloud-based web application, the intention of incoming requests is unknown, not considered and thus each request is serviced with a reply that incurs a fractional cost that is assessed to the cloud consumer. Due to the fact that this vulnerability up until now has not been discussed, determining the overall impact of the threat to the cloud community is difficult at this time. Instead, the focus of this work is to describe and bring awareness to the utility model vulnerability, analyze the risk of a FRC attack for a specific cloud consumer and propose FRC prevention, detection, attribution, and mitigation solutions. The overall goal of this work is to get ahead of this vulnerability before it is inevitably exploited.

1.3 Cloud Utility Pricing Model

The utility model is attractive to a cloud consumer because of the low cost of entry and avoidance of major capital expenses. While convenient, the utility model is not without its risks as the financial liability for resources consumed is unlimited. CSPs such as Amazon EC2 [2] and Rackspace [94] charge $0.12/GB (up to 40 TB) and $0.18/GB respectively for outbound data transfers. As illustrated in Figure 1.1 and based on these pricing metrics, each reply serviced by a cloud application (i.e. the attack target) is assessed a cost to the cloud consumer (i.e. the victim). Requests in sufficient volume can be costly. Malicious use is even more burdensome since the additional run-up in expenses has no associated business value. As it stands today, CSPs do not monitor cloud consumers applications and thus it is up to the cloud consumer to prevent, monitor and respond to such fraudulent behavior [60].

1.4 FRC Attack Description

As evidenced by recent trends in DDoS attacks, attackers are employing the services of botnets with populations of upwards of tens of thousands of compromised hosts and are using these botnets as an attack tool to wreak havoc on the Internet [58, 87]. In order to increase effectiveness and circumvent current detection mechanisms, attackers are moving away from obvious network-layer attacks such as SYN floods and targeting application-layer resources by means of HTTP flooding attacks. This discussion and the description of the FRC attack anticipates a natural evolution of these attacks on metered resources hosted in the public cloud.

In order to describe the FRC attack more precisely, one could consider a time-series visualization of a web server log as seen in Figure 1.2. Reading from bottom to top, the y-axis depicts requests per second and the time-series covers a two-week time period (x-axis). As is common, the modeled web server capacity is sufficiently over-provisioned and this represents a conservative estimate given the capacity of CSP web servers. Superimposed on top of normal web activity are serviced requests from a FRC attack.

As shown in the callout in Figure 1.2, initial attack intensity above normal activity is a region labeled Nuisance Activity because the resultant costs are insignificant for the cloud

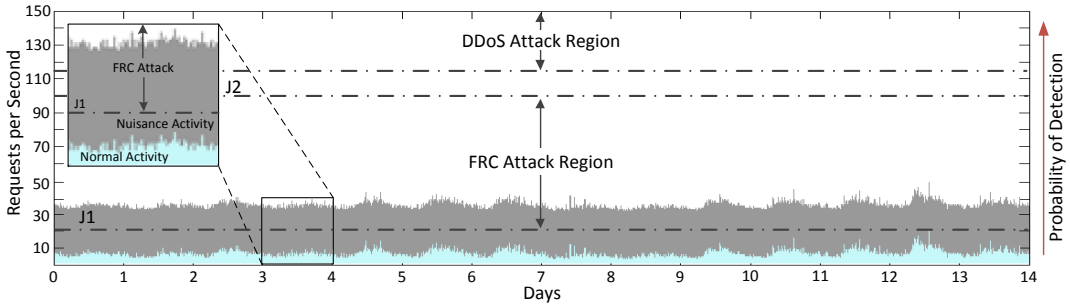


Figure 1.2: FRC Attack Region

consumer. However, as malicious activity intensifies beyond the Nuisance Activity region, the malicious costs to the cloud consumer start to become a matter of concern; this transition point is labeled J1. Malicious activity that exceeds J1 enters into the FRC Attack Region. Within this region bounded by J1 and J2, a FRC attack does not significantly degrade the Quality of Service (QoS) of the web server. With a utility model assigning costs for all data transferred out of the CSP environment, this region is of interest to an attacker who wishes to inflict economic pain. If the attack intensity increases above J2, the request volume will reach a point when the web server QoS starts to significantly degrade. It is at this point that current application-layer DDoS detection and mitigation schemes are effective [58]. An objective of FRC attack mitigation research is to improve detection sensitivity that will push J2 closer to J1, thus narrowing the FRC Attack Region by detecting attacks that are legitimate transactions, but differ in the requestors intent.

Figure 1.3 depicts a FRC attack as a slow-and-low assault or death by a thousand requests. Unlike short-lived DDoS attacks, the duration of a FRC attack could last weeks or months if not detected. Because resources maliciously consumed are additive to that of normal traffic, the aggregate of legitimate and malicious resource use is reflected in a cloud consumers monthly bill.

Availability in the context of this discussion is not a binary measure in which the system is nearly incapacitated at the time of the attack. The technical infrastructure of a website hosted in a CSP environment will have no trouble functioning while a FRC attack is underway. Instead, availability is a long-term consideration defined as the cloud consumer's ability to

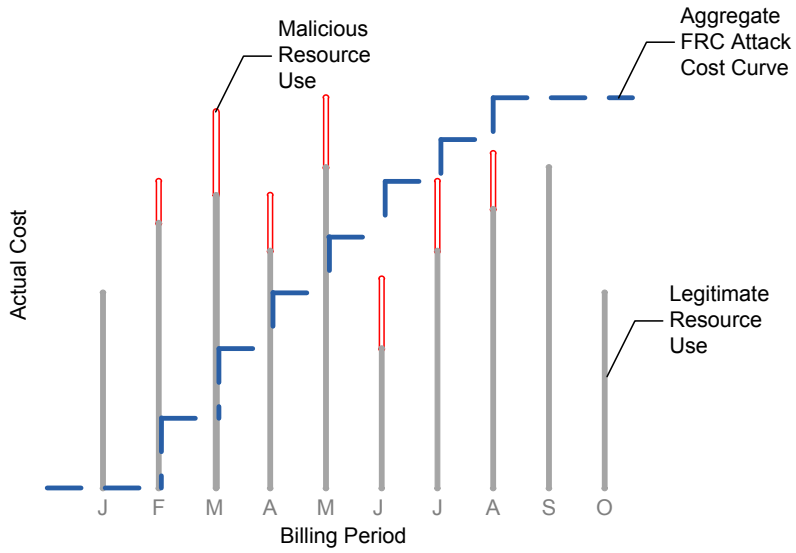


Figure 1.3: FRC Aggregate Attack Cost Curve

withstand the financial consequences of a FRC attack over a prolonged period of time.

Although cloud computing makes for a compelling use case of the utility model, the concept of utility computing is not unique to the cloud model and has been exploited in the past. However, as it stands today, the utility model contains an unaddressed vulnerability for the cloud model and requires the attention of cloud consumers, security practitioners and researchers.

1.5 FRC Risk

Adopting the public cloud model brings with it new and old security risks. A key objective of this article is to bring attention to the risk the utility pricing model introduces by discussing the *likelihood* and *impact* of a FRC attack.

The likelihood of a cloud consumer falling victim to a FRC attack is largely dependent upon the skill level, capacity and motivation of the attacker as well as the exploitability of the utility pricing model. This pricing vulnerability is literally hiding in plain sight as CSPs openly publish their pricing metrics. From a technical standpoint, all that is necessary for an attacker to exploit this vulnerability is to make standard requests for web content that the cloud consumer makes publically available. Although the worst-case threat-source is that of

a large botnet, conceivably any Internet-connected device could perform a FRC attack with a PERL script making HTTP GET requests or through the use of the infamous Low Orbit Ion Cannon - an open-source tool that has fueled recent DDoS attacks [88].

As evidenced by the growing number, capacity, and sophistication of both botnets and DDoS attacks respectively, the worst-case threat-sources undoubtedly possess the skill level and resources to mount a sustained and impactful FRC attack. Thus, the only real factor preventing a FRC attack is the motivation of the attacker. Like those that orchestrate DDoS attacks, the motive of a FRC attacker could range from ego and hacktivisim to monetary gain, extortion, revenge, creating a competitive advantage, and/or economic espionage [113]. If recent history is any guide, those that control botnets would likely perform a FRC attack to promote a political agenda or in support of an ideological viewpoint.

For the victim, the direct monetary impact of a FRC attack is a function of the average request intensity and the duration of the attack. To enumerate one end of the extreme, a week-long DDoS attack launched from a 250,000 node botnet in 2011 peaked at 45Gbps [26]. If such an attack were sustained on a cloud instance at $0.12/GB the resultant costs would have been $0.68/s, $40/min, $2430/h, $58,320/day and $408,240/week. On the other end of the FRC Attack Region, considered the website modeled in Figure 1.2. At an average normal request rate of three requests per second, for a 250,000-node botnet to double the data usage costs of this website would equate to each bot client being responsible for generating two requests per day. Clearly, given the capacity of modern-day networks and computers, the bot clients in this example could significantly increase their daily request quota and multiply the attack cost by orders of magnitude. However, as will be discussed, once a bot clients usage footprint eclipses the expected behavior of legitimate clients, the risk of being identified as malicious greatly increases.

1.6 Defending Against a FRC Attack

Defending against a FRC attack is a significant challenge to the cloud consumer due to the atypical and unassuming nature of the attack. As is the case with most attack risk, the cloud consumer has four primary objectives: prevention, detection, attribution, and mitigation. Each

of these aspects will be considered in this dissertation in the context of the FRC attack.

1.6.1 Prevention

A common way to prevent the exploitation of a vulnerability is to download and apply a patch for it. However, in the context of this discussion, the bug is not a software defect but a common business model deployed by CSPs. Until this vulnerability is actually exploited, the cloud business model is not likely to change, so in lieu of a patch for this vulnerability there are several, albeit limited, prevention options (Chapter 7). While the use of authentication on a target website would significantly reduce the amount of exploitable resources, for this discussion, this website feature is not considered as it is assumed the cloud consumer desires to host public content. Similarly, graphical puzzles (i.e. CAPTCHAS) could be used as a preemptive solution to differentiate humans and zombie computers. However, the use of such a test could be detrimental to the overall goals of a public-facing website as these types of tests will result in a certain percentage of legitimate clients being unable or unwilling to solve such puzzles. Another option would be for the cloud consumer to work with application and content developers to minimize the resource footprint of common or average requests. Limiting the impact of client requests increases the costs for the FRC attacker and the risk of detection. Unfortunately, without a utility model patch these controls will not thwart a motivated attacker. So with limited prevention capability the next line of defense is detection.

1.6.2 Detection

The objective of FRC detection is to be able to determine if malicious traffic consumption is occurring (Chapter 5). Due to the subtle nature of a FRC attack, previous application-layer DDoS solutions that focus on high request intensities will not be suitable for FRC detection [120]. Instead, initial FRC detection approaches focus on behavioral metrics derived from web logs that seek to capture the aggregate web page request choices of a websites client base. Three measures, the Spearman, Overlap, and Zipf metrics have been identified to characterize the accuracy, completeness, and relative proportionality of ranked requests respectively between two adjacent windows of observed logs (e.g. two 3-day windows). Together these three

metrics provide consistent measures with which to describe normal behavior and to perform anomaly detection. Although, for the sake of brevity, empirical results are not presented in this discussion, the conclusion stemming from this work is that an attacker, without knowledge of the training data set (i.e. historical web log), has a difficult time requesting an impactful volume of web documents while adhering to the structure of normal traffic. Thus the proposed methodology is effective for detecting even minor increases in fraudulent web activity, well before the resultant costs are harmful.

The most practical detection approaches are classic. Review the bills over time and determine if they are within an expected range. If not, one possible explanation is fraudulent resource consumption. Log analyzers may help identify outlier application usage that can then trigger an investigation of suspicious clients. A savvy FRC attacker will unfortunately be missed by casual inspection.

1.6.3 Attribution

Attribution in this context is the ability to accurately differentiate legitimate clients from that of FRC attack clients (Chapter 6). Like the previously discussed DDoS detection solutions, current attribution solutions are geared towards detecting malicious clients that consume a significant volume of requests in a very short time. Past works have focused on scrutinizing the increased inter-request (i.e. time between successive web document requests) or inter-session (i.e. time between web browsing sessions) arrival request rates of malicious clients in comparison to that of a profile for normal users [97]. Again, it is contrary to the objectives of a FRC attack for a single attack client to behave in a similar fashion as one participating in a DDoS attack.

The challenge in this research area will be to minimize the number of falsely identified legitimate clients while decreasing the impact of fraudulent clients. The methodology presented in Chapter 6 indicates that normal client behavior can be characterized by client actions such as: request volume per client, web documents requested, and web session parameters (e.g., requests per session and number of sessions). If attack clients, not privy to normal usage activity, exceed a set threshold on these characteristics, they are flagged as malicious. A design

goal of this attribution methodology is to be transparent to the clients and operates under the condition that all clients are innocent until their usage footprint proves otherwise. Limiting the impact of individual clients reduces the overall risk of a FRC attack. It is important to note that this methodology is not rate-based, but instead sensitive to the accumulated requests invoked by an attacker. Therefore, based on the choices made by an attacker, a malicious client could be deemed anomalous after invoking a minimal number of requests.

1.6.4 Mitigation

Reactive solutions rely on accurate detection and attribution (Chapter 7). One must consider the potential for legitimate clients being errantly classified as malicious. As a result, approaches like blacklisting first-time offenders may prove to be heavy-handed. Less absolute mitigation strategies include imposing a back-off timeout to anomalous clients in which requests from an IP address are not all serviced. Similarly, suspicious clients could also be served a graphical puzzle to prove that the client is indeed a human. These reactive approaches are available today and each has its own trade-offs, but with limited detection and attribution solutions available, the deployment and maintenance of such solutions will be challenging.

1.7 Summary

As they are structured today, cloud utility models are vulnerable to exploitation. By allowing any client with access to the Internet to consume resources that are in turn metered and billed exposes the cloud consumer to a risk that is only mitigated by time, detection and accountability. Until recently, this vulnerability has been neglected and there have been no previously known defense strategies. Awareness and understanding are a key means of defense, and this dissertation strives to achieve those goals. Unless utility models are restructured to remove the vulnerability of a FRC attack, research in detection and attribution is necessary to ensure the long-term sustainability of cloud consumers and remove one more impediment that could dissuade organizations from adopting public cloud computing.

To the best of the author's knowledge, there have been no known public acknowledgements of a FRC attack occurring on the public cloud. However, the absence of such knowledge does

not confirm that the utility model vulnerability has not or will not be exploited. As an analog, back in the early 90's, Internet-facing firewalls were new and thought to be sufficient to secure a connected enterprise. However, reality was that attacks were occurring and intrusion detection systems soon pointed out these threats. Perhaps the utility model has been exploited and, as an IT community, we are presently ill equipped to detect its presence or identify its culprits. The only factor preventing a FRC attack is the motivation of the attacker. Through a systematic analysis of the utility model vulnerability, this dissertation seeks to contribute to this research effort.

CHAPTER 2. SECURITY ANALYSIS OF PUBLIC CLOUD COMPUTING

Chapter contains modified content from the following published journal paper:

Idziorek, J. and Tannian, M. Security Analysis of Public Cloud Computing. *International Journal of Communication Networks and Distributed Systems*, Vol. 9, Nos. 1/2, 2012, pp. 4-20, © 2012 Inderscience Enterprises Ltd. (**Invited Paper**)

2.1 Abstract

Cloud computing is in its infancy and continues to evolve. As this evolution proceeds, there are a number of privacy and security concerns emerging from the cloud computing model that need to be addressed before broad acceptance occurs. This chapter is an initial literature survey of cloud computing security, which promises to be a challenging research area. Although cloud computing security research inherits previous research from its elemental technologies, this chapter will limit its focus on surveying cloud computing targeted research. By performing a systematic analysis of the security aspects of the cloud model, this work seeks to succinctly clarify why security continues to be a significant impediment for cloud adoption.

2.2 Introduction

Cloud computing is in its infancy and continues to evolve. Early adopters of cloud computing have recognized the cost savings, convenience, and agility this emerging compute model affords. However, as this evolution proceeds, there are a number of security and privacy con-

cerns emerging from the cloud computing model that need to be addressed before extensive adoption occurs [42]. While much of the initial hype has created an ambiguous characterization of what exactly does and does not constitute cloud computing, this paper will concentrate on the cloud model as described by NIST [77]. The technologies comprising much of this description of cloud computing have been around for some time now (e.g., virtualization, broadband, high- density storage, multi-core processors), however it has not been until recent years that these technologies have all matured to the point where this novel synthesis of these foundational components could be realized. As the cloud computing model matures, so will the body of research that addresses the security concerns. The objective of this chapter is to present a checkpoint of the current state of cloud computing security research by providing a systematic analysis and survey of relevant literature. Although much research has been conducted on the individual components that together support the orchestration of the cloud model, the emphasis of this work will be to concentrate on cloud-specific security research. The aim is to succinctly clarify why security continues to be a highly significant impedance to full-scale adoption.

The paper is organized as follows: Section 2.3 provides the background for cloud computing model. Section 2.4 provides a corresponding threat model. A description of the system of analysis and the subsequent analysis are presented in Sections 2.5 and 2.6 respectively. Finally, the chapter concludes with a discussion in Section 2.7.

2.3 Background

One useful way to understand cloud computing and related security issues is to consider the major components starting with the nature of the cloud host followed by physical resources tenants have in common and finally the connectivity. This work focuses explicitly on public cloud computing where open access is marketed. Three actors will be referenced throughout the chapter. The cloud service provider (CSP) is a company who provides pay-as-you-go services using an infrastructure that is consistent with NISTs definition of cloud computing [77]. The tenant is the CSPs customer who has an application being serviced by the CSP and is responsible for paying for cloud services. The user is the client of the tenant who derives value

from using the cloud-hosted applications.

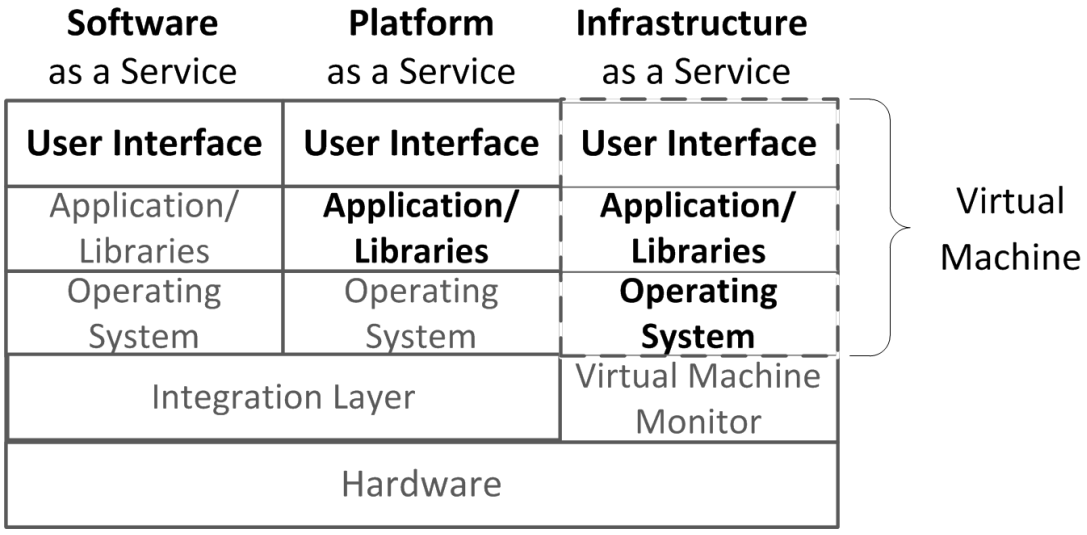


Figure 2.1: Cloud Layer Technology Stack

The cloud host can be viewed as a layered technology stack residing on the physical server hardware. Each of the layers, as shown in Figure 2.1, provides abstraction of resources and functionality to the layers above itself. The completeness and depth of control a tenant has within the stack have been associated with service offerings called software-as-a-service (SaaS), platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS).

In order for cloud computing to be a feasible information technology (IT) offering, CSPs have orchestrated a collection of technologies and investments in a manner that exhibit the properties of economies of scale. In order to achieve these economies, resources are shared among tenants (Figure 2.2). The communications infrastructure between users and tenant applications is common. Multi-tenancy is optimized in order to increase returns on capital investments. The result is that tenants share physical platforms consisting of CPUs, memory, networking and storage. Isolation between tenants is provided through built-in logical controls. Virtualization is one such logical control able to encapsulate tenant operating environments. Large centralized storage arrays are managed with common storage constructs like relational databases for the purposes of holding tenant data.

Connectivity as shown in Figure 2.3 is the last major perspective to consider. The global

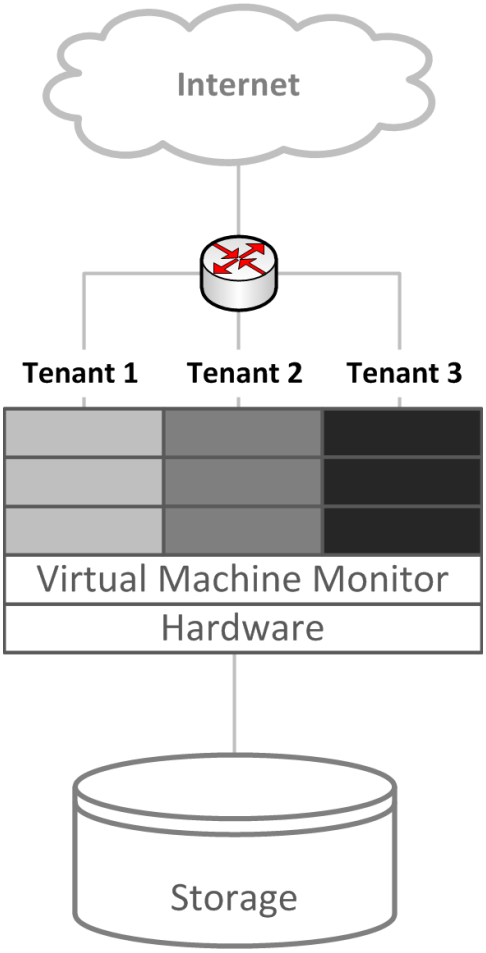


Figure 2.2: Shared Cloud Resources

Internet puts a world-wide market in technical reach of the CSP and users access tenant applications through the Internet. The tenants operational staff manages operations on the Internet. CSP staff can remote in as well. Resilient low-cost service requires high connectivity within the CSP data center in order to facilitate host migration and data replication. Resiliency features and cost reduction are also supported across data centers through high bandwidth intra-connectivity.

2.4 Threat Model

Assessing security and risk for an environment requires a working model of the threats the system under evaluation could potentially experience. Threat modeling of public cloud

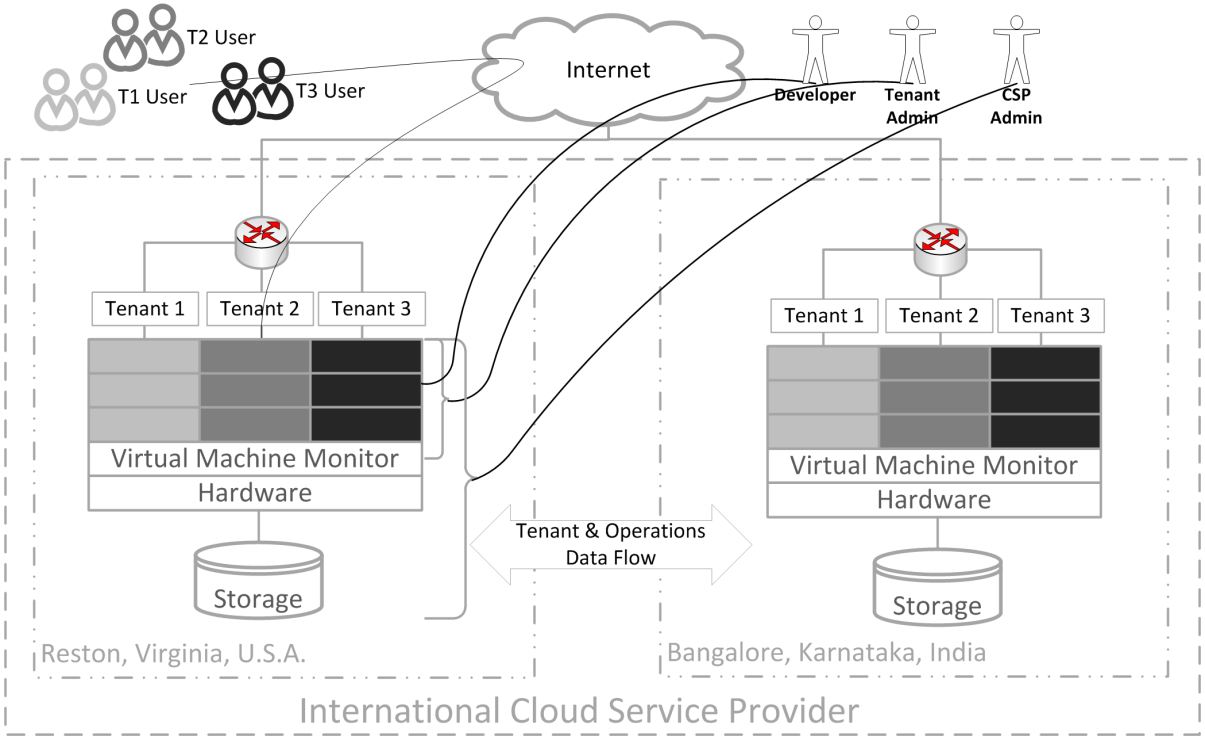


Figure 2.3: Cloud Connectivity

computing in general will encompass a superset of threats to which a specific cloud-hosted application implementation may not be exposed. Please refer to Figures 2.4, 2.5, and 2.6 throughout this discussion. The labelled points represent interfaces through which a threat may exploit access. These labelled points will be referred to as <X.Y-Z> where X.Y is the figure label and Z is a unique designation typically assigned in top-down order within the figure.

When considering threat-sources there are four classes of actors one can consider. The classes are oriented to the origination of their attack. The first class is the Internet originating threat sources <2.5-1>, which includes a long list of usual suspects (e.g., script- kiddy, hactivist, botnet, corporate or government sponsored intelligence operative). The second class and third class of threat sources have the potential to exploit the advantage of proximity and trust. Second is of the semi-privileged actors who operate on behalf of neighboring tenants of the victim (e.g., contractors, employees) <2.5-2, 2.5-3, 2.5-4>. The third and most dangerous class is the fully-privileged actor who operates on behalf of the CSP or the victim (i.e. trusted

insiders) <6.2, 6.3, 6.4>. The last is non-malicious with a wide range of origination points such as operational failure due to natural disturbances, human error or technology faults.

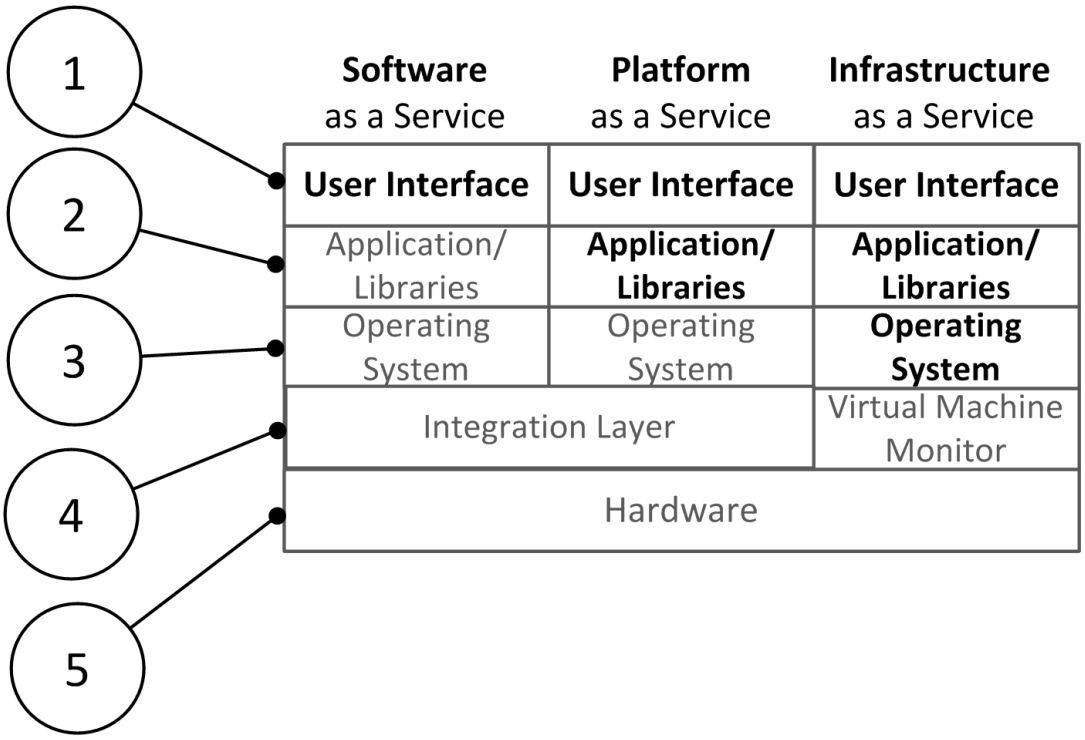


Figure 2.4: Cloud Layer Technology Stack - Threat Interfaces

When considering the conceptual architecture of public cloud computing it appears to present threat-sources with newfound leverage. By moving applications from within private enclaves to the Internet-accessible public cloud, the Internet class of threat-sources has new opportunities to possibly threaten the interface points <2.4-1, 2.4-2, 2.4-3>. The strategic objective of the attackers may vary, but the tactical objectives of exploiting exposed software and configuration vulnerabilities <2.4-1, 2.4-2, 2.4-3> and exhausting or misusing accessible resources can be anticipated. A CSP supplied firewall service may be able to reduce exposure to the ports not relevant to the application, but application exposure is necessary otherwise it is inaccessible to the tenants users.

Although operational failures are not unique to cloud computing, the overall complexity of the architecture that spans from users to the internal cloud implementation has increased. Complexity is the antithesis of operational availability. Greater sensitivity or fragility can be

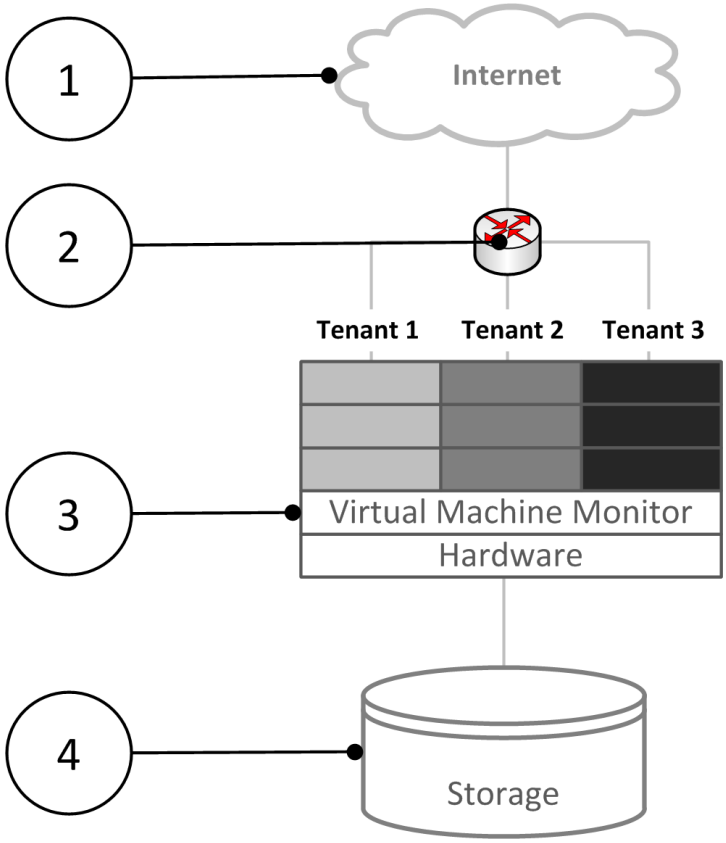


Figure 2.5: Shared Cloud Resources - Threat Interfaces

expected in operations of the public cloud model. Natural events, power outages, DNS root server saturation or cache poisoning, user Internet Service Provider (ISP) connectivity issues, and internal capacity planning mistakes or countless possible configuration errors or system failures within the CSP environment <2.4-1 - 2.4-5, 2.5-1 - 2.5-4, 2.6-4, 2.6-5> threaten user access to cloud-hosted services.

An international CSP raises the specter of geo-politics threatening information flow between facilities. The US Government has proposed an Internet kill switch to isolate US-based information systems from foreign threats [47]. Personal privacy protections instituted in various countries and other regulations limit the physical location of sensitive data storage [41]. The regulatory restrictions and opaque nature of CSP operations threaten organizations with non-compliance if the data storage fault tolerance and archive algorithms within the CSP allow compliance sensitive data to leave the authorized geo-location <2.6-5>.

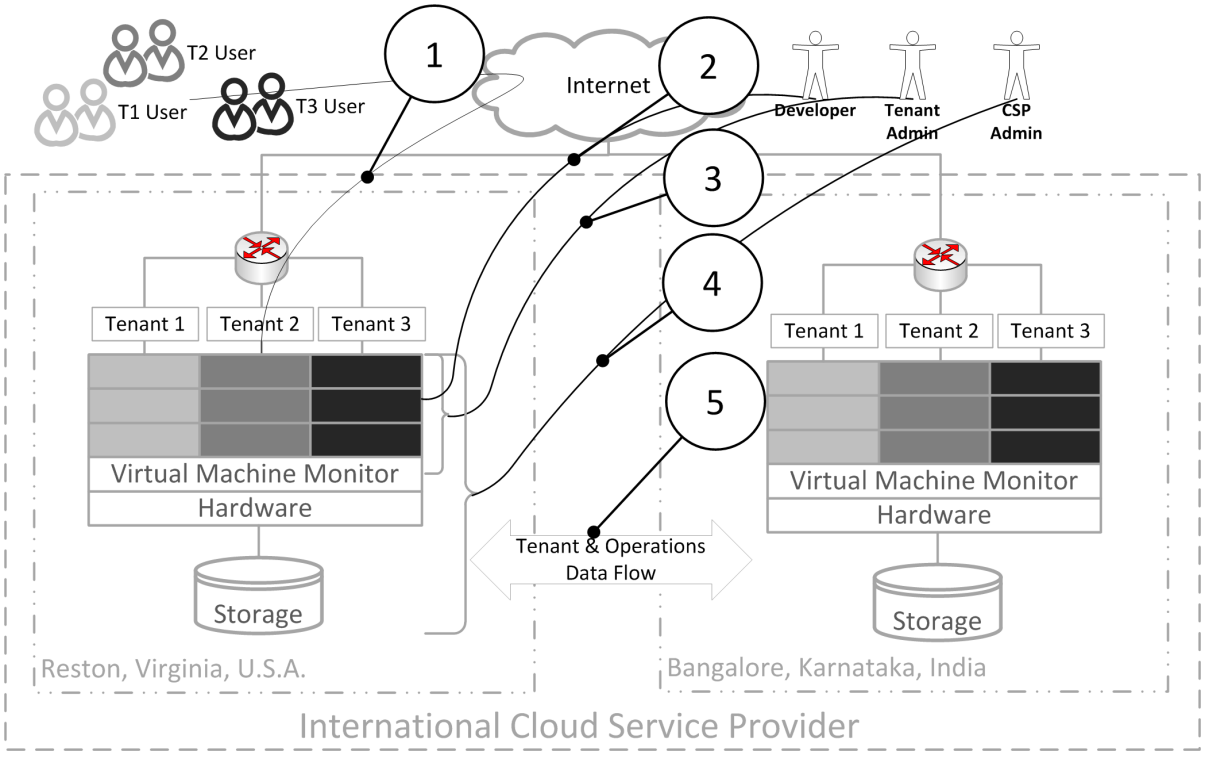


Figure 2.6: Cloud Connectivity - Threat Interfaces

As the literature survey proceeds in Section 2.5, this threat model will help provide context for cloud computing security research conducted to date.

2.5 System of Analysis

Published analysis of cloud computing security is relatively recent. Analysts seem to have waited for definition, implementation and interest to reach a minimum level of maturity. In the past couple of years, at least ten papers have investigated, contemplated and researched broad areas of cloud computing security [22, 48, 49, 54, 55, 57, 61, 62, 63, 66, 90, 96, 99, 101, 106]. The pool of cloud computing specific research is relatively shallow. However, several of these papers endeavored to provide a literature survey [22, 48, 54] using organizing principles like attack surfaces [48] or a compact scheme of: traditional security, availability and third-party data control [22].

This paper reviews the body of cloud computing security literature for completeness of

coverage. A system of orthogonal security principles is used to create a comprehensive review structure to classify existing security research and illuminate gaps in areas of focus. The security field has embraced the tenants of confidentiality, integrity and availability as the qualities secure information systems should attempt to assure. However, such a model is an incomplete set of qualities to consider when securing information and information systems in or adjacent to the public cloud. Donn Parker [89] suggested in 1998 three additional orthogonal elements. These additional elements should prove to be useful for this survey.

The elements we consider are confidentiality, integrity, availability, utility, authenticity and possession. Please refer to Table 2.1 for the definitions of these terms. Since the terms utility, authenticity and possession are not commonly considered, a brief elaboration of these terms will be provided.

Utility on its face appears to be closely related to Availability, but Parker's use of this term avoids any conceptual overlap. Consider the example of digital data on existing paper tape, punch cards, or on 8", 5 1/4" floppy discs. If these media have been preserved, the information stored is available for retrieval. If one ordered a modern computer today, one would find it difficult to request any of these media reading devices and any legacy drivers present in the operating system will have questionable reliability. In this case, there is a lack of device and driver availability, but fundamentally the stored information remains available. Information availability is necessary but not sufficient for it to be useful.

Authenticity and Integrity may also appear to overlap conceptually, but consider that without credible attribution to information origin the information may not suffer from an unauthorized change but from fundamental semantic credibility. Authenticity and data provenance have a lot in common. However, provenance incorporates the notion of the importance of knowing and being able to document the origin and life history of a data object in terms of where it has been. Authenticity is satisfied if there is confidence in the informations authorship or origin as well as if the information conforms to reality or fact. For example, an Internet-hosted software download with an associated unsigned cryptographic hash value has verifiable integrity, however the author or publisher cannot be verified. A verifiable signed cryptographic hash would support authenticity, but would be insufficient to achieve data provenance.

Table 2.1: Parker's Six Security Elements.

Security Element	**Description**
Confidentiality	the attribute of information whereby it has not been exposed intentionally or accidentally to unauthorized entities
Integrity	the attribute of information whereby it has not been subject to unauthorized state change whether intentional or accidental
Availability	the attribute of information and supporting systems whereby they are reliably accessible with minimal delay
Utility	the attribute of information usefulness for a purpose
Authenticity	the attribute of information in which the information is genuine, the assertion of authorship or origin is true, and is overall worthy of trust because it conforms to reality or fact
Possession	the state of information of which an entity has power, physical control or holding of a specific instance of this information, and an opportunity to use the information

Possession is the most fundamental control of an object. The possessor has the opportunity to destroy, alter or utilize the object. Encrypted data held by someone who lacks the decryption key or algorithm remains confidential, but by possessing the encrypted data an opportunity arises to attempt to decrypt the data (e.g., during World War II the allied capture of German submarine communications and Enigma machines). Having a UNIX system password file initially is not a loss of confidentiality. However, by possessing the password file, the only control protecting the passwords is the salted hash. With brute-forcing software, hardware, time and patience, the confidentiality of hashed passwords can be undone. Possession provides the holder the means to violate the other security properties of the information as well. A tenant utilizing a cloud-hosted application has shown a willingness to share or relinquish possession of data

and related information processing.

2.6 Analysis

2.6.1 Confidentiality

Perhaps, the single most significant impediment to the adoption of the cloud computing model results from the lack of confidence in the confidentiality of data processing and storage in a CSPs environment [100]. These concerns are well-founded as CSPs have experienced privacy gaffes in the past that have inadvertently exposed tenants data [107]. When data is moved outside of a trusted domain, data confidentiality issues arise for all of the datas states: data-in-transit, data-at-rest and data-under-processing.

While research on the topic of data storage on untrusted platforms is not unique to the cloud model [76], the proposed use cases for the cloud model (i.e. digital health records) warrant further examination of privacy and confidentiality controls. Initial solutions have been proposed to provide third-party management of encryption keys [127] as well as decoupling access control from the CSP [33].

Even if data is encrypted in transit and it is encrypted at rest, all data processed in the cloud has surely has been decrypted. Recent work in fully homomorphic encryption may provide tenants a means to implement computational processes without exposing the data being processed to the CSP [79]. Fully homomorphic encrypted data can undergo useful transformations without ever requiring decryption by the developed application or the CSP [43].

Interposing technologies used in virtualized environments provide granular views into the activities being performed on the host [31]. Confidential data and any supporting cryptographic key materials are potentially being monitored and logged in the clear as the tenant is utilizing them within processes. Unless an organizations security policy is rewritten to extend authorized entities to include the CSP or make use of specialized hardware and protocols [53], there is no way currently to prevent an unauthorized CSP operator from reconstituting confidential information. Entire sequences of computations can be replayed potentially exposing sensitive activities along with data. A successful implementation of a fully homomorphic system, which

although promising does not seem likely in the near future [79], would have a significant effect on the confidentiality of data stored and processed in a CSP environment.

Although the encapsulation of VMs in a cloud environment is considered to provide logical separation between tenants, recent research has shown that the use of virtualization does not preclude an adversary from engaging in side-channel attacks that threaten confidentiality of a number of facets within a tenants environment [98]. Known as a cross-VM side-channel attack, an attacker on the same physical machine as the victim has the potential to monitor time-shared caches for the purpose of measuring the load of the machine. Based on such knowledge, it has been shown on Amazon's Elastic Compute Cloud (EC2) that an attacker is able to determine whether or not they are co-located on the same machine as the victim (i.e. cloud cartography) as well as estimate the victims network traffic patterns. In the stated examples, although the leakage of such information does not reveal the direct knowledge of data stored or computed in such an environment, it does potentially provide a competitor with the necessary information to inflict potentially damaging DoS attacks or perform inference analysis on activities. Furthermore, sharing resources among fellow tenants opens the door to theft of cryptographic keys via similar cache-based side-channel attacks, covert channels, and keystroke monitoring attacks [98].

Lastly, authentication portals that are exposed to the public Internet raises the threat of unauthorized access by a malicious actor. While brute-force and dictionary attacks and their corresponding mitigation solutions are well-known, research on cloud authentication mechanisms has shown that EC2 was vulnerable to XML signature-wrapping and advanced XSS attacks [112]. If exploited, the attacker would be able to access to a cloud consumer's control panel and thus be able upload and download virtual machine image files and reset administrator passwords to cloud instances. Even though this specific vulnerability has been patched, if history is any indicator, similar types of bugs are ever-present in a CSPs design and lay dormant waiting to be discovered.

2.6.2 Integrity

CSPs provide tenants with simplified computing environments, however the supporting infrastructure and controls are anything but simple. CSPs have not been impervious to Byzantine faults where tenant data has been errantly altered [1]. In addition to the CSP infrastructure, malicious insiders, external bad actors, and malware have the ability to affect the integrity of the cloud model. To mitigate such threats some have suggested that the compute environment in the cloud is the ideal use case for the trusted computing platform [102, 108, 109], while others have explored attestation mechanisms [104] enabling for both integrity verification of a cloud tenants VM and the platform on which it executes. Similarly, others have proposed using a formal language to automate the process integrity verification of both static and dynamic virtualized cloud environments [9]. Although integrity concerns exist for all aspects of a CSPs platform much of the early attention has concentrated on the integrity of the data stored in the cloud and the integrity of the virtual compute environments (i.e. IaaS) [50].

Initial analysis of integrity vulnerabilities in cloud storage systems find that CSPs are able to provide integrity safeguards for data as it is transferred in and out of the cloud but lack the ability to provide such integrity verification between transferring phases as data is stored on a physical medium [35]. The absence of such controls in current day cloud storage offerings brings forth issues of not only integrity but also repudiation between tenants and CSPs. Related mitigation efforts propose third-party controls that enable the verification of integrity for dynamic data stored in the cloud [119] as well as security services that provide intermediate integrity protections between applications and cloud storage providers [110].

In order to achieve efficiencies and to minimize cost, many CSPs rely heavily upon virtualization technologies. CSPs utilize virtualization as an abstraction of computing resources that enables several virtual machines (VMs) leased by multiple and distinct tenants to reside on one physical server. The key to abstracting hardware resources is the use of the virtual machine monitor (VMM) also referred to as a hypervisor (Figure 2.1). The VMM acts as a broker between each VM and the underlying hardware platform comprising of CPU cycles, storage, memory, and other computing resources. Reliance on virtualization by CSPs enables unique

security advantages but also creates inevitable trade-offs introducing new security vulnerabilities [50, 93].

In a non-virtualized computing environment, host-based security mechanisms are accessible by the user leaving these mechanisms vulnerable to attack, disablement, and misconfiguration. Transferring security services from within the guest operating system and placing them in the VMM mitigates these threats while offering advanced capabilities to monitor the integrity of a virtualized compute environment. The process of monitoring VM behavior is known as virtual machine introspection (VMI) and security research in this field has increased significantly along with the resurgence of virtualization [38]. A few of the security controls that can be migrated are passive intrusion detection monitoring of VMs, malware detection and secure logging. Those security services that take advantage of their introspective position are able to perform security anomaly detection by either modifying processes or providing a decision framework for secure code execution [80]. Whether CSPs offer such services by default or as add-on security-as-a-service features, the ability to monitor the integrity of the guest operating system at a layer outside control of the tenant and across an entire virtualized enterprise provides the potential for security functionality that individual tenants are not capable of providing for themselves.

With the pervasive use of VMMs in cloud environments, the integrity of the VMM has become a key security focus. Just like traditional operating systems, VMMs are also software implementations and thus leave the door open for similarly exploitable vulnerabilities. Threats include but are not limited to interposition attacks [64] to either observe or alter VM data flows or destroying the entire computer environment [44]. The tactical advantages of hosting security mechanisms in the VMM may be lost if an attacker manages to obtain control of the lowest layer in the cloud technology stack (Figure 2.1). Whoever controls the VMM has a clear advantage to either ensure or disrupt the integrity of the cloud environment [38].

2.6.3 Availability

One of the cloud computing models most significant contribution is its potential to make vast sophisticated resources available to tenants. However, a few high-profile incidents have caused much disdain for the cloud model. The severity of such outages is exacerbated by the

fact that availability is lost for all parties involved. Not only are users unable to gain access to cloud-based resources via the Internet, but also tenants are unable to gain local access to their resources, because they are also dependent on Internet connectivity. While initial focus has centered on specific outages from known threat sources, research efforts have started to explore vulnerabilities that are either created or enhanced by the cloud model.

High-profile CSPs like Amazon, Google, and Microsoft have experienced undesirable down-time in the past [84] due to power outages [16], weather [30], and technical problems [15]. However, the vulnerabilities that have accounted for these past losses of availability are not unique to the cloud model and are equivalent for virtually all IT data centers. As inferred from many CSPs service level agreements (SLAs), outages are to be expected. Amazons EC2 [2] and Microsoft's Azure [121] provide availability guarantees of 99.95% uptime. While a single two-hour outage might grab news headlines, over the course of a month this loss of availability is well within the stated CSPs SLAs. If a mission-critical service is unable to tolerate periodic and inevitable periods of unavailability, it is likely that these services are not suitable for the cloud.

Despite fault-tolerance resource planning and novel features such as horizontal scaling, CSPs have not been resilient to DDoS attacks. In one specific instance, a tenants cloud-based web server was denied service for more than 19 hours [78]. While DDoS attacks are not unique to the cloud model and have been experienced by Internet-facing services for quite some time, the increased reliance on Internet-facing services raises the risk for such attacks. Further complicating the issue is that tenants must not only accept the risk of such attacks on their resources but must also account for the collateral damage resulting from attacks targeting collocated tenants.

In addition to traditional threat sources, the architecture and shared resources that are inherent to the cloud model increase the threat-surface for those seeking to deprive availability. As a cornerstone of the cloud computing business model, multi-tenant provisioning results in multiple tenants information processing needs being serviced by common resources. Relating to availability, resource consumption by each tenant draws from a common pool of CPU, RAM, network infrastructure, and disk storage to name a few. In such an environment, the attack

has more control over the compute environment (i.e. IaaS service platform giving the attacker the most control and the SaaS service platform giving the attacker the least control), thus enabling the attacker to disrupt availability of service for fellow tenants. Recent research has focused on an attackers ability to map the cloud network topology [73] and also the attackers ability to place a VM on the same physical host as their victim [98]. In the former example, by mapping out the network topology, an attacker can perform flooding attacks on limited network bandwidth between subnets and thus denying service to tenants on the same subnet. In the latter example, by placing a VM adjacent to that of the victim, the attacker is able to observe the victims network activity. This provides the attacker with the insight of when peak loads occur and when to inflict a DDoS that will cause maximum damage.

Transferring information processing or data into the cloud that was traditionally hosted on private networks increases operational availability exposure and introduces new vulnerabilities attackers can exploit. In spite of this downside, the cloud model enables the acquisition of enormous amounts of computing resources and each tenant must answer the question of whether or not the cloud model can provide higher availability in comparison to in-house hosting.

Missing from initial threat modeling and research is the consideration of availability in respect to the long-term financial viability of a cloud tenant to operate in the cloud. While unexpected costs in the cloud can originate from mismanaged or unanticipated usage [115], such costs can also be inflicted by an attacker seeking to exploit the utility pricing model that governs the usage of resources in the public cloud. This gap in threat modeling and research is the problem this dissertation seeks to address.

2.6.4 Utility

Utility of information should be considered on a wide range of information abstractions. What is an application or virtual web server platform, but information that performs with a purpose? In many cases these infrastructure and application platforms are as essential as the data being processed in order to obtain utility from the data [117]. Although information utility is not a commonly discussed term, lock-in is of great concern [13, 22, 117]. Lock-in is the circumstance that data, application design or implementation security controls require signifi-

cant transformation efforts in order to operate these elements in a different CSP environment or back in-house. The concept of lock-in is not unique to cloud computing when one considers difficulty related to moving on-premises applications to different platforms (e.g., legacy applications). The lack of standard interfaces, APIs, security control architectures, databases, and implementation parameters cause much of the lock-in [13, 22, 117]. Beyond the data processing aspects, proprietary data formatting is also a concern [22].

Exit strategies may be a means to avoid lock-in [13, 96]. Whether the language is the SLA [96] or another contractual document [13] the format of the data should be agreed upon for when the tenant wishes to leave the CSP. It is unclear whether there will be value lost in terms of metadata or structure when the data is exported to a neutral or widely accepted format.

One strategy in avoiding lock-in would be for a tenant to self-manage as much of the cloud services environment as possible. In other words, the more the CSP does for the tenant the more the CSP unique methodologies, applications, programming platforms and interfaces bind the tenant to the CSP [13]. This strategy defeats the value of cloud computing for some tenants, because the necessary skills, labour and software licenses increase the upfront and ongoing costs of entry for a tenants desired services.

Use of predicate and homomorphic encryption has been suggested to improve control and confidentiality [22], but it may also be a means to improve utility. High assurance implementations of these technologies will require widely recognized standards of use (e.g., AES is a usage standard for the Rijndael cipher) as well as vetted implementations. As a result, there is likely to be greater portability of programs written to use these technologies as well as that data format thus allowing for portability across CSPs or for internal processing. Key management will be an important consideration, because any purpose requiring the data to be in clear-text may be thwarted if the keys are lost or unavailable.

2.6.5 Authenticity

Authenticity is generally desirable for all information types. Authentic academic transcripts, health records, tax filings, market prices for corporate shares or bonds, corporate financial

records clearly demand authenticity since decisions are dependent on this information being true to fact. There is a greater tolerance for doubt in authenticity for data like today's weather forecast or whether a staff meeting has been rescheduled, but one would desire authenticity if verifiability or accountability could be easily achieved. When control of data storage and information processing is entrusted to a third-party how does one assure the authenticity of the information and services in their control?

Authenticity is an emergent property whereby trust in an information object is a result of demonstrable discipline that results from properties like: transparency of operations, governance of information services, change control, separation of duties, auditing, authentication and accountability. Outsourcing information processing and business services like managed data centers, payroll or credit card processing is a fairly well-understood business practice. What makes public cloud computing more challenging is: the purposeful use of multi-tenancy; dynamic broad geographical dispersion of information storage and processing; a highly competitive cloud services market; proprietary cloud architectures, management systems and service offerings technology; unrestricted customer base and global Internet user access. Verifiable discipline in an environment as described can dissuade conservative customers from adopting public cloud computing.

Analysts and researchers have made some progress in identifying and addressing these authenticity challenges. Researchers [22] have raised the issues of lack of transparency in cloud operations, frequent change in cloud services, the challenges of auditing a dispersed environment, the heightened need for appropriate client authentication, the governance issues surrounding the CSP practice of outsourcing services to a subcontractor and the lack of accountability for potential loss of intellectual property. Some solutions have been suggested to address cloud risks like: using tenant controlled middleware that ensures authentic data transactions for data stored with an untrusted storage provider [110]; implementing trusted computing to address the lack of transparency by providing high-assurance remote server attestation [22]; and [62] reports that VMWare has released an API that provides hypervisor transparency.

Progress to address authenticity has been made; however, until authenticity becomes an explicit objective of information security practitioners it will remain a second order issue.

2.6.6 Possession

When analysts explore the root cause for the concerns over public cloud computing, possession is the root of all control concerns. Loss of control not only jeopardizes the other five elements of security, but is also a source of practical and legal challenges in its own right. It would seem that the fundamental need for information protection and the desire for cheap plentiful information processing are irreconcilable. For some tenants this may never change, but research into this challenging problem may help others identify a means to safely adopt public cloud computing.

As for practical problems, when a third-party possesses data how does one assure proper deletion of information especially when some providers leverage their enormous collection of tenant data for mining purposes [22]? These cross-purposes between the tenants needs and providers revenue sources raise trust issues. Although not discussed by others, possession raises residual data handling concerns after a tenant abandons an application. The risks may not be borne by the tenant, but by the subjects of or the end-users who were using the abandoned records. Although CSP departures from the market have been documented [22], the prospect of mergers and acquisitions within the CSP population or by other firms looking to diversify may raise the potential for a tenant to be in competition with its CSPs parent company. In other words, a competitor may come into possession of a tenants data. Combine this prospect with lock-in, the tenant is in a difficult position.

As for legal problems, which will vary by jurisdiction, there are privacy and other regulatory concerns. Concerns about regulatory compliance have been raised by [55] and [22], and these authors associate the root issue to be possession. However, if authenticity is verifiable the regulatory concerns regarding possession may be mitigated in cases where regulation does not stipulate geo-location. Privacy can be viewed from two perspectives. One is the privacy of the tenant and the other is privacy of the user. US legal standing of information held by a third-party is murky. Currently, the US Fourth Amendment does not appear to apply to the cloud thus there appears to be no need for judicial review for governmental access to data stored in the cloud. Antiterrorism and criminal pursuits enabled by the US PATRIOT Act and

the Stored Communications Act ease the burden of proof by the government and may limit the cloud tenants awareness that their information has been obtained by the government [41]. Geo-location of data is a regulatory constraint for individual-privacy laws as specified in European Union Directive 95/46/EC and the associated national legislation for each of European Unions member countries. The controller as designated in Directive 95/46/EC is the tenant who is operating an application obtaining personal privacy information. Although the responsibility for privacy protection is initially upon the tenant, it may be possible to share responsibility with the CSP if the CSP is deemed to be influencing the means by which privacy information is being processed. Per [22], CSPs are responding by allowing tenants to target the geo-location of their data.

One suggestion to address cloud possession issues is to enable information to be self-protecting [13], which essentially moves the security perimeter to be around the individual data object. In addition, the use of predicate encryption and homomorphic encryption [22, 43] may help alleviate disclosure-risk related to data remnants and trust issues related to CSP ownership. Others have sought to provide solutions that would enable a cloud tenant to verify the geo-location of data stored in the cloud [8].

Safe relinquishing or sharing of information possession is an open problem where potential solutions have value far beyond cloud computing.

2.7 Discussion

Research papers on public cloud security have been reviewed using a framework of six security elements. The practical reality is that tenants need to account for all six security elements simultaneously. Each tenant must decide the significance and priority these elements have with respect to their needs. For comprehensive security within the cloud, researchers and practitioners need to work on addressing all security elements. Such complex evaluations and decisions that follow will be costly.

Beyond the technical challenges of identifying research problems and solutions, the cloud computing subject is plagued by realistic problems. Each CSP has implemented a cloud in their own unique fashion, which stymies efforts to generalize technical topics and solutions.

The competitive nature of cloud computing discourages openness and access to the underlying layers of abstraction. To complicate the issue further, cloud computing and related terms (e.g., SaaS) remain ill-defined thus challenging efforts to define the target of evaluation with precision. Moreover, the target of analysis (i.e. the individual cloud) is undergoing constant change. A finding discovered today may be moot in a matter of days or weeks. This amorphous target being presented by CSPs only promotes security through obscurity. Without standards with respects to terms, functionality, protocols and interfaces, meaningful security research in cloud computing that is comprehensive will be difficult to conduct. Unless standards are embraced and the gaps like those raised in the Analysis Section are addressed, security will be an ongoing obstacle to public cloud computing despite its alluring advantages such as cost savings.

CHAPTER 3. FRAUDULENT RESOURCE CONSUMPTION ATTACK

Chapter contains modified content from the following published conference paper:

Idziorek, J. and Tannian, M. "Exploiting Cloud Utility Models for Profit and Ruin." *In Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing (CLOUD '11).* Washington, DC. 4-9 July 2011. pp. 33-40, © IEEE 2011. (**Acceptance Rate 36/196 = 18%**)

3.1 Abstract

This chapter introduces and discusses an attack on the cloud computing model by which an attacker subtly exploits a fundamental vulnerability of current utility compute models over a sustained period of time. Internet-accessible cloud services expose resources that are metered for billing purposes. These resources are subject to fraudulent resource consumption that is intended to run-up the operating expenses for public cloud service customers. The details and significance of this attack are discussed as well as initial detection and attribution methodologies and there respective experimental results. This inaugural work investigates a potentially significant vulnerability of the cloud computing model that could be exploited from any Internet-connected device. Because the explored exploit is simply a matter of making well-crafted web transactions that only differ in intent but not in content of the attacking client, such attacks are challenging to differentiate and thus this attack may be difficult to detect and mitigate.

3.2 Introduction

Computing services that were traditionally hosted on organizations' servers and networks are being outsourced to third-party Cloud Service Providers (CSPs). Transferring sensitive data and computing operations outside of a trusted environment raises obvious concerns of confidentiality, integrity, availability, authenticity, utility, and possession for all aspects of a CSP's service platform (Chapter 2). Initial focus, research, and threat modeling has concentrated on both the confidentiality and integrity of data stored and computed in the cloud. Absent from these works is an analysis of the external threat sources that have the ability to directly or indirectly affect the availability of cloud-based services and exploit the integrity of the billing model that governs this emerging computing model.

Despite uncertainty and unknown security vulnerabilities, early adopters of the cloud computing model have utilized cloud-based resources for a number of services including search engines, web hosting, content delivery, and application hosting [3]. Adopting cloud services remains an unknown risk for many customers; however, initial customers have recognized the benefits of reduced overhead of capital expenses and the attractiveness of the more novel cloud infrastructure features such as horizontal scaling and the pay-as-you-use billing model. An obvious threat that CSP customers face is the loss of availability. Although high-profile CSPs like Amazon, Google, and Microsoft have experienced undesirable downtime in the past [84] due to Distributed-Denial-of-Service (DDoS) attacks [78], lightening [30], and technical problems [12], these CSPs offer service availability guarantees in the range of 99.9% to 99.95% uptime [2, 45, 121]. The vulnerabilities that have accounted for past losses of availability are not unique to the cloud model and have analogs in virtually any web-based service. With an increased reliance on web-based services, including the outsourcing of applications that were traditionally desktop-centric services, an organization raises the likelihood of DDoS attacks affecting those operations hosted in the cloud.

While semantic and flooding DDoS attacks are well known and the associated risks are well researched [87], this work will explore a subtle attack more akin to an application-layer DDoS attack. The threat-source considered in this paper is an attacker who seeks to fraudulently

consume bandwidth of web-based cloud services that in turn incur a financial burden on the cloud consumer. Utility computing is particularly vulnerable to an attack by which the attacker seeks to exploit the utility pricing model in order to financially harm the victim.

The attack scenario depicted in Figure 3.1 illustrates a vulnerability of the cloud utility model. A botnet consisting of potentially thousands of bot clients under skillful control of the botmaster can consume cloud resources by mimicking legitimate client behavior. The aggregation of these requests is the problem this work seeks to address.

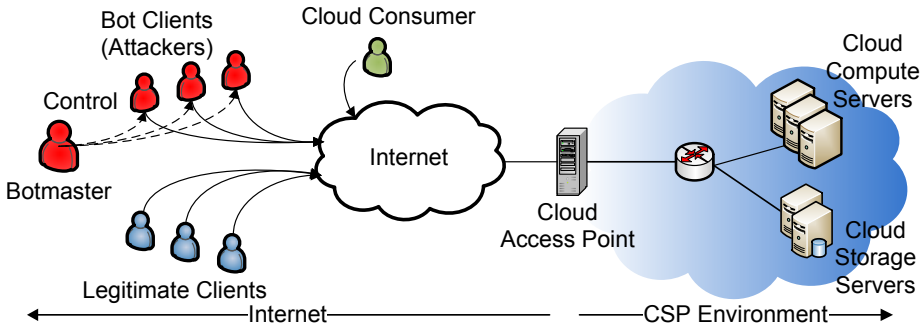


Figure 3.1: Cloud Attack Network Model

In this chapter, two initial methodologies to mitigate Fraudulent Resource Consumption (FRC) attacks on metered web resources are presented. The first is a detection methodology that applies the properties of Zipf's law to the analysis of aggregated user web consumption patterns. The second approach is an entropy-based attribution methodology that detects anomalous behaviors in request dynamics for individual attackers. The experimental results demonstrate that both methodologies show promise of being effective at detecting FRC attacks.

The rest of the chapter is organized as follows. Section 3.3 provides the background for the cloud computing model. A detailed description of the FRC attack is described in Section 3.4 and the detection and attribution methodologies as well as experimental results are presented in Section 3.5. Related areas of work are analyzed in Section 3.6. Finally, future work and the conclusion are presented in Section 3.7.

3.3 Background

The attack described in this paper is not unique to the cloud model and could be carried out in similarly hosted environments that makes use of utility pricing such as content distributed networks. Despite the fact that this type of attack is not specific to cloud computing, the cloud model does provide a well-documented and practical application of the utility computing model to demonstrate such an attack. To better understand this attack and to provide context for remainder of the chapter, this section provides a brief overview of public cloud computing and the utility model.

3.3.1 Cloud Computing

When broken down, cloud computing is a specialized distributed computing model. Building upon the desirable characteristics of cluster, grid, utility, and service-oriented computing, cloud computing introduces a unique complement of features to create a new computing paradigm [37]. The technologies comprising much of cloud computing have existed for a while (e.g. virtualization, broadband, high-density storage, multi-core processors); however, it has not been until recent years that these technologies have all matured to the point where a synthesis of these elements could be realized. One of the novel characteristics of the cloud model - as compared to past computing offerings - is that consumers have the ability to self-provision computing resources. This allows a cloud consumer to quickly establish an Internet-facing web presence. Furthermore, CSPs offer such services at attractive prices by way of a utility pricing model as made possible by virtualization and economies-of-scale.

To keep the terminology straight throughout the rest of the chapter, the following roles are defined:

- *Cloud Service Provider (CSP)* - The CSP (e.g. Amazon, Microsoft, or Rackspace) offers client-provisioned and metered computing resources that can be leased for flexible time durations.
- *Cloud Consumer* - The cloud consumer is a person or organization that employs the services of a CSP and is financially responsible for any and all resource consumption.

- *Client* - The client is a legitimate user that requests services offered by the cloud consumer.
- *Attacker* - The attacker is a malicious user that fraudulently consumes resources offered by the cloud consumer.

3.3.2 Utility Compute Pricing Model

The utility model, by which cloud services are offered, enables the attractive payment model of pay-as-you-use. Customers pay only for the resources they consume and only for the time that they consume them. The flexibility that this model facilitates is advantageous to a cloud consumer because of the low cost of entry and avoidance of major capital expenses. Table 3.1 represents a subset of the direct costing metrics established by Amazon's Elastic Compute Cloud (EC2) platform [4]. Although cloud computing makes for a compelling use case of the utility model, the concept of utility computing has origins that date back to time-sharing on mainframe computers.

Table 3.1: EC2 Pricing Metrics for a Large Linux Instance Residing in Northern Virginia (as of January 2012)

Standard Compute Instance	
Large Linux	$0.34 per inst. hour
Data Transfered In	
All Transfered In	$0.10 per GB
Data Transfered Out	
First 1 GB per Month	$0.00 per GB
Up to 10 TB per Month	$0.15 per GB
Next 40 TB per Month	$0.11 per GB
Next 100 GB per Month	$0.09 per GB
Over 150 GB per Month	$0.08 per GB

As seen in Table 3.1, the cost of computing in the cloud is billed in units of Cost-Per-Hour (CPH) consumed and derivatives of Cost-of-Data-Transferred (CDT) in and out of the CSP's environment. The total cost of computing in the cloud model can then be modeled as follows:

$$Total\ Cost = CPH(hours) + CDT(bytes) \tag{3.1}$$

Equation 3.1 represents the cost of cloud resources as a generic model that allows for the analysis of cloud consumer costs independent of a particular CSP and their respective cost metrics.

3.4 Fraudulent Resource Consumption (FRC) Attack

This section provides a conceptual foundation for the FRC attack. In order to provide a comprehensive explanation, the target of the attack, the threat model, an attack description and an exploration of the direct costs associated to this attack are provided.

3.4.1 Target

For the purposes of this work, the target of this attack is a website or web application hosted in a third-party CSP environment. The CSP will generate revenue by providing hosting services on a utility model basis. In this service environment, resources consumed by clients result in a direct cost to the cloud consumer.

The characteristics of the websites considered in this paper are those that have been designed to serve predominantly public web content that is accessible to Internet users. Although the use of authentication on the site would significantly reduce the amount of content readily available to the general anonymous public and thus potentially restrict the amount of exploitable resources, this website feature is not considered for it is assumed the cloud consumer desires to host public content. It has also been assumed that the target website is hosted in an environment in which the web server is properly configured, patched and is protected behind a firewall that conforms to a well-considered information security policy and employs best practice filtering techniques.

An additional characteristic is that the website does not make use of reverse-Turing tests [58] to differentiate humans from zombie computers. The use of such tests is detrimental to the overall goals of the cloud consumer, as these types of tests will result in a certain percentage of legitimate users choosing not to solve the puzzles as well as preventing search bots from indexing the website's content. Therefore employment of such techniques to restrict access to public-facing web content is regarded as excessive and not considered in this work.

In an effort to simplify the experimental design used to assess the chosen detection and attribution techniques, the scope of HTTP protocol request methods is limited to HTTP GET requests. All HTTP request methods consume resources on the server and supporting network, but do vary by degree of consumption. These consumption distinctions would be necessary to consider if a precise cost calculation is desired, but for the purpose of this initial work on the FRC attack the focus is on the general relation between client actions and direct cost to the cloud consumer.

3.4.2 Threat Model

The threat sources for the target described in Section 3.4.1 are the common threat sources typically associated with Internet applications and services, such as a script-kiddy, hacktivist, extortionist and a person performing information warfare for commercial or government purposes [28]. The universal access that the target's service provides enables remote access from any Internet-connected threat source.

With a black market of hackers or botnets for hire [58], a threat-source is not required to be capable of performing the attack themselves. Whoever performs the attack will require sufficient compute resources and bandwidth to implement a sustained and significant resource utilization attack, which is fairly given the current computer technology, capacity, and bandwidth.

In the past, Internet attacks were generally regarded as being less financially motivated and driven by attackers need for self-fulfillment, political motivation, fun, or proof of skill [72]. Today, however, cyber criminals have been moving towards making a profit. The motivation of the hired attacker in this threat model is purely financial and the attacker benefits directly from either from a service fee or from an extortion fee paid by the victim. The original threat-source, who hired the botnet master, achieves their objective by decreasing the economic health of the victim.

In this threat model, the attacker will factor in time for attack completion, attack success likelihood, and attack detection as key variables as resources are allocated and as an attack methodology is chosen. By understanding the utility models published by CSPs, the attacker can determine what transactions or actions will cost. Although optimal attack strategies are

outside the scope of this paper, a wise attacker will construct an attack to consume significant amounts of resources but will stay clear of extreme resource consumption to avoid detection. The attacker will craft proper functional transactions in order to ideally exercise as many billable resources per transaction as possible while remaining undetected. Unlike a SYN flooding attacks that seeks to consume available socket resources by forming numerous incomplete TCP connections, fully-established application-layer HTTP connections are much more effective at consuming large volumes of resources while remaining undetectable by current signature and anomaly-based detection mechanisms.

The threat source attacks by generating web traffic consistent in comparison to legitimate traffic. However, it is assumed that the threat source does not have the ability to access historical records/logs from the victim's servers nor the ability to insert a traffic-logging device in front of the victim's web-based services. Although collusion with an inside person would do away with the previous assumptions, these fairly realistic restrictions prevent the attacker from creating statistically indistinguishable traffic patterns in comparison with legitimate traffic seen by the site. From the victim's perspective, malicious and legitimate traffic are interwoven and only differs in intent not content.

3.4.3 Attack Description

As evidenced by recent trends in DDoS attacks, attackers are employing the services of botnets consisting in size of upwards of tens of thousands of compromised hosts and are using these botnets as an attack medium [58, 87]. To increase effectiveness and circumvent current detection mechanisms, attackers are moving away from network-layer attacks such as SYN floods and attacking application-layer resources by means of HTTP flooding attacks. This paper and the description of the FRC attack anticipates a natural evolution of these attacks on the computing resources metered for utility pricing as found in the cloud computing model.

The recent emergence of cloud computing and its attractiveness has raised the prospect that current utility model structures may prove to be a significant vulnerability to cloud consumers. The attack is simply a matter of making properly formed and seemingly legitimate requests for application services in sufficient quantity that expenses accumulate over time to a level that

is unsustainable for the cloud consumer. One key objective of the attacker is to blend into the noise of legitimate activity so that their malicious resource consumption is undetected by current measures and their activities remain unimpeded. What makes the attack unsustainable for the victim is that the victim's business objectives for the cloud-based services are not achieved regardless of the disproportionate amount of expenses paid.

In order to describe this attack more precisely, one could consider a continuum of Malicious Resource Utilization (MRU) as seen in Figure 3.2. Reading from bottom to top, the y-axis on the left-hand side of the figure depicts a gradual increase of resource utilization (%) for a busy NASA web server over a two-week duration of time (x-axis). The y-axis on the right-hand side of the figure denotes the number of requests per second experienced by the web server. Because of the historical nature of the data set, a direct mapping of this relationship is not known, but is depicted to represent a conservative estimate given the capacity of modern day web servers.

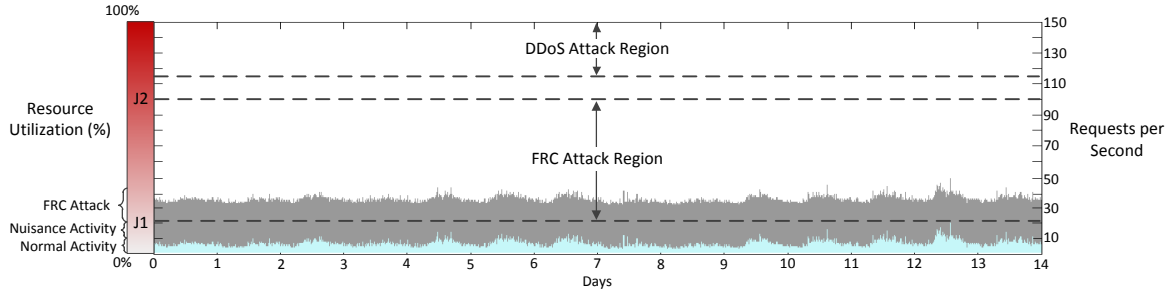


Figure 3.2: Malicious Resource Utilization Continuum

Initial attack intensity above normal activity is a range labeled nuisance activity because the resultant costs are insignificant to the cloud consumer. However, as the malicious activity intensifies beyond the nuisance activity range, the malicious costs to the cloud consumer start to become a matter of concern. This transition point is labeled J1. Malicious activity that exceeds J1 enters into the FRC Attack Region. Within this region, a FRC attack is neither nuisance activity nor does it significantly degrade the QoS of the web server. With a utility model assigning costs for all resources consumed, this region is of interest to an attacker who wishes to inflict economic pain. If the attack intensity increases above J2, the aggregate resource consumption will reach a point when the QoS starts to significantly degrade as the increase

resource utilization results in an increase in system response latency. It is at this point detection like, as stated in [56, 58, 120], current application-layer DDoS detection and mitigation schemes will be activated. The transition point between the inability and ability to detect malicious resource consumption is denoted as J2 on Figure 3.2. The initial objective of the FRC attack research is to improve detection sensitivity and push J2 closer to J1 by improving detection of attacks that appear as legitimate traffic and transactions, but differ in the requestor's intent.

Availability in the context of this discussion is not a binary measure in which the system is nearly incapacitated at the time of the attack. The technical infrastructure of a website and its provider will have no trouble functioning while the FRC attack is underway. Instead, availability is a long-term consideration defined as the cloud consumer's ability to withstand the financial consequences of such an attack over a prolonged period of time. Unlike a short-lived DDoS attack, the duration of a FRC attack is intended to last weeks or months. As shown in Figure 3.3, a FRC attack is similar to a slow-and-low approach in which the costs for resources maliciously consumed are additive to that of normal traffic. The challenge the FRC attack raises is that of the detection of malicious activity that blends in with normal behaviors with the intention of subtly exploiting the resource sensitivity of current utility pricing models in order to incrementally increase operating costs thus inflicting financial damage.

3.4.4 Direct Cost of a FRC Attack

DDoS attacks have always resulted in some from of financial loss for the victim. Whether directly or indirectly, the victim experiences loss when legitimate clients are not able to access revenue generating services, productivity is halted, or the service or corporation's reputation is damaged. Not until resource consumption was directly billable to the consumer, as is the case in the cloud model, was it possible to associate a direct monetary cost to the compute and networking resources consumed during a DDoS attack. The focus of this work can be seen as a much more subtle variation of a DDoS attack. Because of this, both FRC and DDoS attacks on services hosted in the cloud model can be modeled in terms of resources consumed and the resulting monetary loss for the victim.

Regardless of the motive, an attacker and cloud consumer must consider the same set of

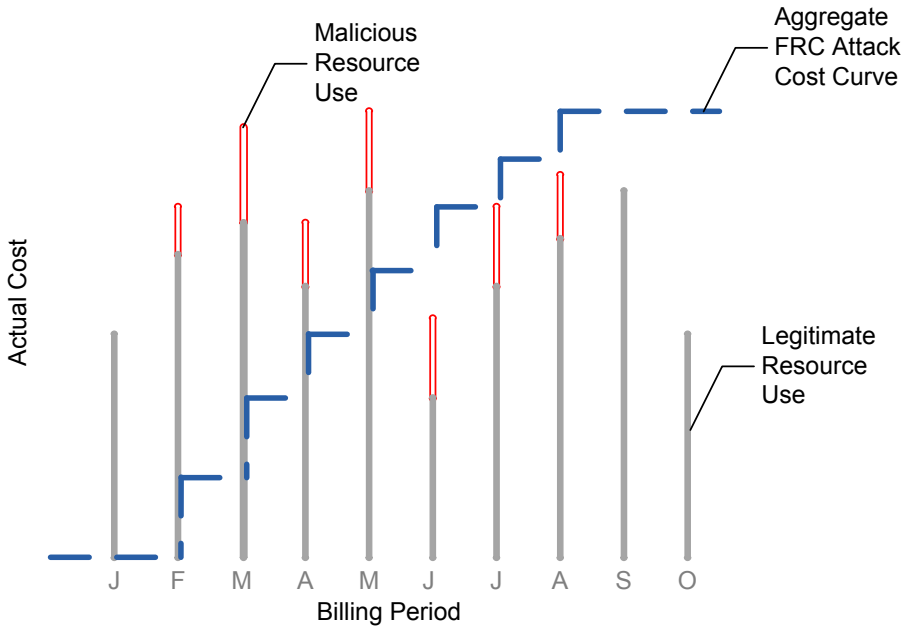


Figure 3.3: FRC Attack Cost Curve

parameters in order to calculate the expected or actual cost of resources consumed in the cloud. The one distinction between these calculations is that the attacker only considers resource usage in excess of normal activity while the cloud consumer must account for the total cost of all resources consumed despite the intention of the requestor.

From the parameters presented in Table 3.2, the total amount of data transferred into the cloud consumer's environment via HTTP GETs over a given time period (γ) can calculated as $D_{in} = \rho \cdot \delta \cdot \gamma \cdot \theta$ and the corresponding amount of data transferred out during the same time period can be calculated as $D_{out} = \phi \cdot \delta \cdot \gamma \cdot \theta$. The subscripts "N" and "A" are used to differentiate between normal activity and resources consumed as part of a FRC attack respectively.

$$Base\ Cost = f(D_{in_N}, D_{out_N}, \gamma, \mu) \tag{3.2}$$

$$Total\ Cost = f(D_{in_N} + D_{in_A}, D_{out_N} + D_{out_A}, \gamma, \mu) \tag{3.3}$$

$$FRC\ Attack\ Cost = Total\ Cost - Base\ Cost \tag{3.4}$$

Table 3.2: CSP Bandwidth Cost Parameters

Parameters	Description
Number of clients (δ)	Number of distinct clients requesting resources
Average resource size (ϕ)	Average size in bytes for each outbound resource request
Average request size (ρ)	Average size in bytes for each inbound request
Request frequency (θ)	Requests per time period
Time duration (γ)	Time elapsed between the beginning and end of an observed period
Cost model (μ)	CSP pricing model
Cost function f(D_{in},D_{out},γ,μ)	Cost of resource consumption

Tiered costing models such as the one used by Amazon's EC2 (Table 3.1) require that the FRC Attack Cost be calculated as the Total Cost of all activity minus the Base Cost. Data consumed during a FRC attack is additive to that of normal activity and the FRC Attack Cost cannot be accurately calculated without knowing the Base Cost.

The consequences of a FRC attack may be best illustrated by quantifying the cost of an attack based on a realistic scenario. Presented next is a scenario of a FRC attack on a web service hosted on Amazon's EC2. Proposing hypothetical attacks in conjunction with the FRC Attack Cost highlights the potential impact FRC attacks can have on a web-based resources hosted in the cloud.

3.4.4.1 Scenario - EC2

Google calculates that the average web page currently found on the Internet is 320KB in size [95]. Assuming this is the average page size of the cloud consumer's website, which consists of multiple distinct pages, an attacker is able to consume on average 320KB per each primary HTTP GET request and its subsequent secondary in-line requests. In this scenario, normal activity is assumed to be 1TB of data per month resulting in a Base Cost of $153.45. At the

rate of requesting one page per minute every minute for a month, a single attacker is able to consume approximately 13 GB of data. Applied to the FRC Attack Cost equation, this attack alone results in a charge of $2.04 for data transferred in and out of the cloud consumer's environment.

Table 3.3: Single Attacker Scenario

Parameters	**Value**
Number of clients (δ)	1
Average resource size (ϕ)	320KB
Average request size (ρ)	1KB
Duration (γ)	31 days
Request frequency (θ)	1 req/min
Cost model (μ)	Amazon EC2
FRC Attack Cost	$2.04

The cost accrued from a single attacker at the given rate in Table 3.3 would likely be characterized as nuisance activity found below J2 as established on the MRU continuum (Figure 3.2). Although a non-zero cost, this malicious resource consumption is likely to blend in with the noise of an average monthly service bill.

Table 3.4: Multiple Attacker Scenario

Parameters	**Value**
Number of clients (δ)	1000
Average resource size (ϕ)	320KB
Average request size (ρ)	1KB
Duration (γ)	31 days
Request frequency (θ)	200 reqs/day
Cost model (μ)	Amazon EC2
FRC Attack Cost	$283.81

In the next attack variation, the number of active bots in the attack is increased and each bot attacks with a request frequency of 200 transactions per day. These adjustments change the

magnitude of the attack from nuisance activity to an attack intensity above J1 and thus into the FRC attack region. The consequence of the utility model vulnerability as presented in Table 3.4 is a bit more apparent. If the attack were distributed throughout the course of a month, even at 200 requests per day from a 1000 bots, the attack does not begin to significantly degrade the QoS of the website (4 x 1.2 GHz 2007 Xeon CPUs, 7.5 GB RAM, 850GB storage) - assuming the system was designed with sufficient performance headroom for the normal activity.

As seen by the attack scenarios in the case study of Amazon's EC2, a FRC attack can inflict a noticeable financial burden on the cloud consumer over time. As the number of attacking resource consumption bots increases, damaging attacks can be mounted without being able to associate a significant usage footprint to a single client, as compared to what was seen in the single attacker scenario. Because current detection efforts are focused on excessive amounts of HTTP requests over a short period of time [56, 87, 120], as is the case in DDoS attack and flash crowds, it is likely that FRC attacks will go undetected.

One important observation worth mentioning is the relative cost of conducting the FRC attack. Although there may be costs associated with lease time and/or the corresponding attack intensity, the amount of bandwidth and compute cycles for a bot to perform 200 requests per day (i.e. on average, one request every 432s) over 31 days is a mere fraction of what a reasonably modern computer is capable of producing. Based on this presented scenario, there should be sufficient performance margin for many more requests per second for each bot being devoted to the attack or other attacks concurrently. While the FRC attack described in this work is presented as a subtle attack, this does not preclude the much more obtuse DDoS attack from exploiting the same vulnerability of the cloud model. For a cloud DDoS victim, not only would they be hindered by the loss of availability, they would also be financially responsible for the bandwidth necessary for an attacker to mount such an attack.

3.5 Detection and Attribution Methodologies

In this section, initial methodologies for both FRC detection and attribution will be discussed. In the context of the described FRC attack, detection refers to being able to differentiate increases in normal aggregate traffic from that of a FRC attack. Attribution, on the other

hand, refers to the ability to accurately identify malicious clients from that of legitimate clients. Because the realm of FRC attack detection and attribution have been largely unexplored, the methodologies explored in this chapter are an initial attempt to push J2 down the Malicious Resource Consumption Continuum (Figure 3.2) towards J1. By pushing J2 to a lesser intensity, attackers will need to sustain their attack longer to achieve the same cost impact or increase the number of bots needed to mount a successful attack. As performance of these initial detection methods are evaluated, the common concerns of computational efficiency and overall detection latency are less of a factor for detection and attribution solutions due to the long duration of a FRC attack. Although ultimately desirable, such efficiencies are not an immediate concern in the short-term, especially if attack clients initiate attack footprints on the order of 200 requests per day. Unlike click fraud or DDoS attacks, which require solutions that have optimized computational efficiencies to deal with these events in a very timely manor, accuracy is the overwhelming key factor for the mitigation of FRC attacks. Just as it is assumed that a web server under a FRC attack will have no problem providing a high quality of service, it is also assumed that such a server will be able to calculate the presented metrics in sufficient time to make a timely detection/attribution decision.

3.5.1 Detection using Zipf's Law

The first question to answer when attempting to mitigate a FRC attack is whether or not a system is under attack. Although such knowledge does not directly lessen or prevent a FRC attack, such knowledge is crucial when considering attribution trade-offs and for performing risk analysis. If it can be accurately determined that an increase in traffic volumes is the result of legitimate client behavior, then it is counter productive for the cloud consumer (i.e. FRC defender) to deploy a mitigation solution that will unnecessarily introduce false positives into the system and hinder or reject legitimate clients.

The objectives of this section are to investigate an application of the properties of Zipf's law [132] for detecting anomalies in web request logs as well as to discuss experimental design and empirical results of applying Zipf's law to the detection of FRC attacks. This initial detection methodology holistically explores a web server log in terms of web document frequency

and the relative document popularity (i.e. document rank).

An effective anomaly detection approach could be a useful indicator for fraud in metered web services. In the past, Zipf's law has been used for detection of anomalous patterns in large data sets such as in detecting blog spam [81] and accounting fraud [52]. With respect to applying Zipf's law to the web, Zipf-like distributions of web logs have been used for modeling and formalizing web caching models and algorithms [14]. A key property of Zipf's law is that it allows for the broad analysis of very large sets of data.

Consider a web server log that contains user-generated request records for a website consisting of N distinct web pages. Let f_i be the frequency of requests for the i^{th} of N web pages and let r_i be the rank of that document. To construct a Zipf distribution and given a web log, let all the web pages and their respective request frequencies be ranked in descending order of the popularity where the most frequently referenced page is assigned the rank of one and the i^{th} page is the i^{th} most popular page. If Zipf's law holds, the frequency of a request for the i^{th} most popular web page is inversely proportional to the rank of the page and is represented as follows:

$$f_i \propto \frac{1}{i}(i = 1, ..., N) \tag{3.5}$$

If K is the number of occurrences for the most frequently requested web page (i.e. rank of one), then given the rank of any web document, its frequency can be calculated as follows:

$$f_i = \frac{K}{i} \tag{3.6}$$

Shown in Table 3.5 is the rank and corresponding frequencies for a fictional website whose user base conforms exactly to Zipf's law and therefore abides by Equation 3.5. When the corresponding rank (r_i) and frequency (f_i) are plotted on a log-log scale (Figure 3.4), the observed slope (ψ) of the best-fit line is negative with a value of unity.

While the synthetic example conforms exactly to Zipf's law, it has been shown through research efforts [14, 56] that web page requests for an actual website instead generally follow a Zipf-like distribution where the frequency of a request for the i^{th} most popular page is a

Table 3.5: Theoretical Zipf Distribution Rank and Frequency

Rank(r_i)	Frequency (f_i)	Log10(r_i)	Log10(f_i)
1	10000	0.00	4.00
2	5000	0.30	3.70
3	3333	0.48	3.52
4	2500	0.60	3.40
5	2000	0.70	3.30
6	1667	0.78	3.22
7	1429	0.85	3.15
8	1250	0.90	3.10
9	1111	0.95	3.05
10	1000	1.00	3.00
11	909	1.04	2.96
12	833	1.08	2.92
13	769	1.11	2.89
14	714	1.15	2.85
15	667	1.18	2.82
16	625	1.20	2.80
17	588	1.23	2.77
18	556	1.26	2.74
19	526	1.28	2.72
20	500	1.30	2.70
21	476	1.32	2.68
22	455	1.34	2.66
23	435	1.36	2.64
24	417	1.38	2.62
25	400	1.40	2.60
26	385	1.41	2.59
27	370	1.43	2.57
28	357	1.45	2.55
29	345	1.46	2.54
30	333	1.48	2.52

power-law function such that:

$$f_i \propto \frac{1}{i^{\psi}} \tag{3.7}$$

To provide a more accurate illustration, Table 3.6 provides the rank and respective frequencies from the NASA dataset (introduced in Chapter 4 and Figure 3.2). As it can be seen in the corresponding plot in Figure 3.5, common distributions of web page requests are not truly consistent with Zipf's law as Zipf's law states that ψ is unity. However, web page requests tend to be consistent with a more general Zipf-like distribution that allows ψ to be close to but not unity.

Using the resulting slope of the regression line from a Zipf-like distribution as a detection metric, experiments seeking to exploit the consistencies in Zipf-like distributions for a particular website were performed using datasets from web request logs produced by Iowa State Univer-

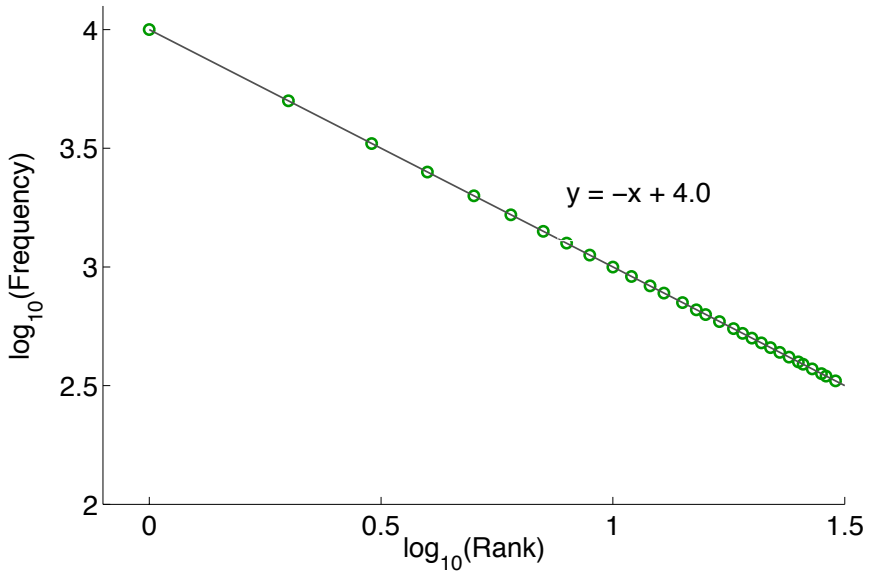


Figure 3.4: Synthetic Zipf Distribution

sity's public web server (www.iastate.edu) over the course of nine consecutive weeks from late 2010. The first week of data served as the training data set and was used as the model of the site's normal web page access patterns. The subsequent weeks served as test data sets as well as the background traffic in which synthetically generated attack patterns were inserted.

The evaluation consisted of generating an attack through synthetic construction of malicious requests and then interleaving these requests within the test data sets. The advantage of synthetic generation is that it enables expedient testing of a number of scenarios and request patterns of attackers in order to test the limitation of the Zipf-like distribution based detection approach. In order to emulate a FRC attack, synthetic requests were generated to consume a percentage of web-based resources above that of normal activity. The synthetic request construction methodology assumed that the attacker had *a priori* knowledge of the magnitude of normal web page requests for a given site. Although contrary to Section 3.4.2, such an assumption was made to allow the attacker to generate more challenging attacks that start at J1 and increase in intensity up the MRC (Figure 3.2) continuum. The web request pattern generated was the union of normal and attack web requests. For each week of test data, five

Table 3.6: Empirical Zipf Distribution Rank and Frequency

Rank(r_i)	Frequency (f_i)	Log10(r_i)	Log10(f_i)
1	9577	0.00	3.98
2	1026	0.30	3.01
3	983	0.48	2.99
4	867	0.60	2.94
5	824	0.70	2.92
6	813	0.78	2.91
7	768	0.85	2.89
8	757	0.90	2.88
9	707	0.95	2.85
10	693	1.00	2.84
11	651	1.04	2.81
12	594	1.08	2.77
13	578	1.11	2.76
14	502	1.15	2.70
15	497	1.18	2.70
16	475	1.20	2.68
17	471	1.23	2.67
18	423	1.26	2.63
19	363	1.28	2.56
20	362	1.30	2.56
21	361	1.32	2.56
22	338	1.34	2.53
23	330	1.36	2.52
24	328	1.38	2.52
25	311	1.40	2.49
26	306	1.41	2.49
27	285	1.43	2.45
28	267	1.45	2.43
29	266	1.46	2.42
30	265	1.48	2.42

synthetic attack data sets were constructed in addition to the original data set. In the attack data sets, the attack modeled uniformly random page requests totaling 5%, 10%, 20%, 30% and 40% more requests than that of the original data set for a given week.

Based on the hypothesis that two weeks of web traffic would produce statistically similar Zipf-like distributions, the detection methodology compared the training data set and each of the test data sets using Analysis of Covariance (ANCOVA). The first step was to compute the Zipf-like distribution for the training data set and each test data set. The second step was to determine if there was a statistically significant difference between the slopes of the two Zipf-like distributions under examination. This required the computation of a linear regression line for each distribution. The slopes of the respective linear regressions were compared with the statistical hypothesis that the slopes are the same. If analysis indicated that the slopes were significantly different, then this result was interpreted to signify that a sufficiently large web page request pattern anomaly took place thus performing as the fraud detection threshold.

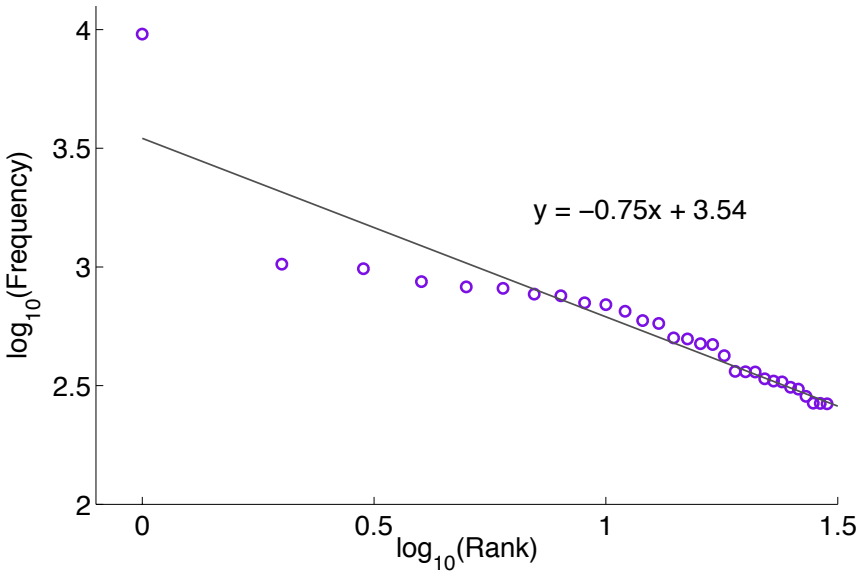


Figure 3.5: Actual Zipf Distribution

Given the lack of specific attacker attribution, additional detection techniques, as presented next, would also need to be deployed.

Table 3.7: Zipf Detection Confusion Matrix

Actual/Predicted	Positive	Negative
Positive	95%	5%
Negative	11%	89%

To measure the effectiveness of this application of Zipf's law for the 48 tests performed, a confusion matrix of detection results is given in Table 3.7. Preliminary empirical results show that the methodology produces a False Positive Rate (FPR) of 5% and a False Negative Rate (FNR) of 11%. These initial results would lead one to believe that this broad analysis methodology appears to be an effective way to examine large sets of web logs for the purpose of detecting possible fraud motivated access patterns in web request logs. However, this proposed detection methodology has its limitations. Namely, as will be addressed in Chapter 5, the tail of a Zipf-like distribution tends to deviate from Zipf's law and thus provides inconsistencies that

lead to errant classifications. Furthermore, this presented methodology only considers a single aspect of a web log. Applying the lessons learned from this initial work on FRC detection, Chapter 5 will explore a much more complete solution that also consider the completeness and accuracy of compared web logs in addition to a revised application of Zipf's law.

3.5.2 Entropy-Based Attribution

Once a web log has been successfully determined to contain FRC attack requests, the next logical question to answer is which of the clients in the web log are malicious and which are legitimate. Similar to the previously discussed work on FRC detection, this section provides an preliminary methodology for FRC attribution.

Web server user interactions can be modeled as a series of successive requests grouped together to form web sessions over a given time period. Requests are human-initiated events utilizing HTTP protocol commands such as GET in order to retrieve web content. A series of related requests generated by a specific user form a single web session. Web request logs provide no indication of when sessions end or start. In order to establish user sessions within the request logs it was assumed that a 900 second or greater pause between consecutive requests of a single user indicated an end of a previous and start of a new web session [67]. Session length was defined as the number of web documents requested during a web session. While previous research has examined the use of statistics derived from web request logs such as source IP address frequency [70] or request inter-arrival times [87] within a web session, the focus of this detection methodology is to model individual user behavior by analyzing the entropy of session lengths generated by an individual over a fixed duration of time in order to detect FRC attacks on the utility model.

Publicly available web request logs, commonly used for flash crowd and DDoS detection research [70, 87], were used as normal activity (i.e. training datasets) to conduct the entropy-based detection experiments. One data set originated from activities observed by a busy ISP web server over a two-week period [23]. The second collection is two months of web logs from a busy NASA website [82].

The attacks within the experiments were modeled as a botnet that consisted of 500 distinct

bots that generated for each bot, on average, a total of 200 malicious requests over a given week. This fixed volume of traffic represented 46-113% more traffic than the original volume for a particular week. Motivated by the threat model established by Oikonomou [87] in which attack bots randomly requested web sessions lengths between 1 and 50, the malicious web sessions used for the purposes of these experiments were composed to resemble much more realistic usage patterns by generating uniformly random session lengths between 1 and 15 web content requests.

The hypothesis of this initial FRC attribution solution is that randomly generated session lengths, as previously described, deviate sufficiently from a profile of normal user behavior in such a way that detection and attribution of attacking bots is possible using an entropy-based detection scheme.

The proposed detection methodology includes both a learning stage and a detection stage. The learning stage involves computing a standard of entropy of normal session lengths for users that invoke five or more sessions as observed in a web request log designated as the training data set. The detection stage consists of computing entropy of session lengths for each unique user and comparing the entropy result to the standard. If a user's session length entropy is outside the standard, the user is designated as malicious.

Entropy has been used in many detection contexts, including application-layer DDoS detection [118, 124]. If the probability that a discrete random variable X takes on a value x_i, given by $p(x_i) = P(X=x_i)$, then the entropy of session length for session j is H_j composed of the n events is defined as follows:

$$H_j = -\sum_{i=1}^{n} p_i log_2(p_i) \tag{3.8}$$

The entropy of session lengths for a given data set is a random variable H that exhibits the properties of a normal distribution. Each weekly web request log data set served as a training data set while the remaining seven data sets represented potential FRC attacks. One advantage of training on each week is that if flash crowds were present they would not be errantly detected as malicious as is the case with some DDoS approaches. Only relative anomalous behavior is flagged as malicious.

To distinguish normal user behaviors from anomalous behaviors, a tolerance interval bounding 90% of the assumed usual traffic (γ=0.90) was calculated with 95% confidence (α=0.05) using a two-sided tolerance interval $\bar{h} \pm k_2 s$ where $\bar{h} = \sum_{i=1} h_i/n$ is the mean of the entropy of session lengths for the n respective clients, s is the sample variance and k_2 is:

$$k_2 = \sqrt{\frac{(N-1)(1+\frac{1}{N})z^2_{(1-p)/2}}{\chi^2_{\gamma,N-1}}} \tag{3.9}$$

For this two-sided tolerance interval, γ is the critical value of the chi-square distribution with N-1 degrees of freedom. This test considers α percent of the sample population with $z_{(1-p)/2}$ as the critical value for the normal distribution with confidence (1-p)/2.

This metric was applied as the standard of entropy to the test data sets that potentially contained attack traffic mimicking a FRC attack. Experimental results for FPR and FNR are shown in Table 3.8 and Table 3.9 respectively. These tables summarize the findings of the 56 experiments utilizing the two months of NASA web request logs that were segregated into 8 weekly subsets. The ISP Web logs experiments produced the following results: week 1 as the training set FPR: 4.0%; FNR: 1.2%, week 2 as the training set FPR: 9.1%; FNR: 0.6%.

Table 3.8: Attribution False Positive Rates (%)

Train/Test	Wk1	Wk2	Wk3	Wk4	Wk5	Wk6	Wk7	Wk8
Wk1	-	4.4	5.1	8.2	6.1	7.7	7.1	11.6
Wk2	8.0	-	6.5	9.2	8.2	13.9	5.7	14.2
Wk3	4.0	4.4	-	7.1	6.1	7.7	7.1	12.2
Wk4	4.0	3.7	4.4	-	6.1	7.7	5.7	10.3
Wk5	4.0	3.7	4.4	4.1	-	7.7	5.7	10.3
Wk6	1.3	2.5	2.9	3.1	4.1	-	4.3	7.1
Wk7	2.7	3.1	3.6	3.1	4.1	4.6	-	7.7
Wk8	2.0	1.9	2.9	3.1	4.1	1.5	5.7	-

Initial experimental results show that entropy-based detection of session length variation is potentially an effective means to detect and reduce the effectiveness of FRC attacks. Like most attributions schemes, the methodology as it is currently devised can be defeated. By increasing the number of attacking bots and decreasing the amount of sessions produced by each bot, the attacker can achieve their objective and moderate the amount of entropy the attack produces

Table 3.9: Attribution False Negative Rates (%)

Train/Test	Wk1	Wk2	Wk3	Wk4	Wk5	Wk6	Wk7	Wk8
Wk1	-	3.4	5.8	4.8	4.4	3.4	3.2	4.0
Wk2	7.0	-	6.4	4.8	6.8	8.8	4.6	6.4
Wk3	5.4	3.2	-	4.6	6.0	3.8	2.6	5.2
Wk4	3.6	3.6	3.8	-	3.0	3.6	3.2	4.2
Wk5	6.8	4.6	6.4	6.8	-	4.2	7.2	5.4
Wk6	9.2	10.2	9.0	6.8	6.0	-	7.0	7.6
Wk7	7.2	5.0	7.4	4.8	6.8	5.6	-	7.0
Wk8	5.4	6.0	7.0	6.4	5.0	6.0	5.4	-

thus remaining within the tolerance limit. Furthermore, if an attacker consumes less than five web sessions, they will also go unidentified. In a practical context with an attacker lacking the insight on usage patterns, staying within the tolerance limit can be difficult to judge by the attacker.

While the proposed attribution scheme exhibited limited success, like the initial FRC detection solution proposed, it is not a general solution and requires further refinement. Chapter 6 will improve upon this approach by considering a more apt and encompassing set of client web usage characteristics.

3.6 Related Work

This section provides a survey of related work that has bearing and similarities with that of a FRC attack. Because the mitigation and description of the FRC attack has been largly unexplored and unaddressed, the related bodies of work are derived from many areas.

3.6.1 Economic Denial of Sustainability

The notion of an Economic Denial of Sustainability (EDoS) attack has been previously discussed in non-academic forums. The term was first presented on a blog posting by cloud computing security professional Christopher Hoff [51] and has since been discussed in similar contexts [24, 91]. Hoff describes the EDoS attack as a purposeful manipulation of a utility pricing model that exceeds the economic means of a cloud consumer. This description of the

vulnerability of the utility model in the cloud has served as a key motivation for this work. This work on the FRC attack is a refinement that considers a more subtle threat model and subsequent detection and attribution.

Taken at face value, an EDoS attack is an attack category that encompasses DoS/DDoS, click fraud, and FRC attacks. Whether directly or indirectly, the majority of DoS/DDoS attacks have had economic motivations. The victim suffers financially by degradation of service capabilities, by payment of an extortion fee, loss of reputation, or elects to spend additional resources to bolster defense capabilities. Likewise, click fraud has always been an economically motivated attack in which the attacker directly benefits from surreptitious requests. In comparison to the previously stated attacks, the FRC attack, as discussed in Section 3.4, has deep-rooted economic motivations in which the attacker uses a slow-and-low strategy to perform an attack that leads to an EDoS over a prolonged period of time. Although Hoff first proposed the concept of a FRC-like attack, this work formalizes a concrete understanding of the utility model vulnerability, proposes both detection and mitigation solutions, and considers this problem in the context of similar technologies, research, and mitigation strategies.

3.6.2 Application-Layer DDoS

There have been a number of key works that have explored application-layer DDoS attacks that have resulted in potential detection and mitigation techniques that may be applicable to FRC attacks. Although the attacker's objectives and request intensities of DDoS attacks are significantly different from that of a FRC attack, this particular body of work is relevant because both attacks employ similar attack methods to mimic the behaviors of legitimate clients.

Within this body of work, some have sought to distinguish flash crowds from DDoS attacks [56, 120, 124]. It was found that the number of overall client requests in a flash crowd was proportional to the number of users [124] and that flash crowds do not exhibit higher per-client request rates [56]; both are behaviors that differ from DDoS attacks. While significant, these findings are not applicable to the detection on a FRC attack. The FRC attack is fundamentally different from the behaviors of flash crowds as these events are composed of dramatically increased amounts of *normal traffic* over a short period of time.

Due to the nature of DDoS attacks - a large amount of requests during a short period of time - detection and attribution of malicious clients has focused on statistical methods for detecting such behavior. Ranjan et al. [97] implemented a DDoS defense that consists of a suspicion assignment technique that is reliant on an abnormal increase in the inter-arrival request frequency for individual users and an increased session inter-arrival frequency over all clients to be successful. Oikonomou et al. [87] proposed a technique to model normal user behavior by exploring increased metrics for session and request inter-arrival times as well as the average inter-arrival rate per client session. Finally, Jung et al. [56] used similar methods of flagging offending attackers by monitoring per-client request rates. The effectiveness of each of these three solutions is contingent on the individual malicious clients performing requests at a significantly higher rate during a DDoS than that of normal traffic. This is contrary to a FRC attack in which a malicious attacker would only need to make on the order of 200 requests over a given day that would not be detected by these DDoS detection methodologies.

Others have analyzed request semantics of users [87, 124], which appears *prima facie* to be a promising area for future work that could be used in tandem with the presented FRC detection methodologies.

To be fair, the approaches discussed in this section were not designed nor analyzed with respect of a FRC attack by their respective authors. Until now, the FRC attack has not been a research subject. Future examination and experimentation will answer the question of whether or not these detection methodologies are limited to application-layer DDoS attacks or whether they possess qualities that can be adapted for the purpose of detecting significantly reduced but still malicious consumption of web content.

3.7 Future Work & Conclusion

Future research on this topic will focus on shifting the J2 point on the MRU continuum ideally to where J2 = J1 - ϵ or in other words the FRC attack has been relegated to below nuisance activity. While research in the area of application-layer DDoS attacks has been focused on mitigating attacks when the QoS of the target begins to degrade, methodologies and approaches from this body of work serve as a promising catalog of techniques on which to base

attribution and detection methodologies that are appropriate for more subtle attacks like the FRC.

Utility models as they are structured today for cloud computing are vulnerable to remote exploits. By allowing any user with access to consume resources that are in turn metered and billed to the cloud consumer, exposes the cloud consumer to a risk that is only mitigated by time, detection and accountability. Until now there have been no previously known mitigation strategies. Awareness and understanding are a key means of defense, and this chapter strived to achieve those goals. This chapter provided a thorough description of the FRC attack and described initial detection and attribution methodologies that may contribute to or motivate solutions to mitigate such attacks. Unless utility models are restructured to obviate the FRC attack, research in mitigating the FRC attack is necessary in order to ensure long-term sustainability of cloud consumers and remove one more impediment that has dissuaded organizations from adopting utility model based computing services like cloud computing.

CHAPTER 4. TRAFFIC GENERATION

Chapter contains modified content from the following published conference paper:

Idziorek, J., Tannian, M. and Jacobson, D. "Modeling Web Usage Profiles of Cloud Services for Utility Cost Analysis." *In Proceedings of the 2011 Winter Simulation Conference (WSC).* Phoenix, AZ. 11-14 Dec. 2011. pp. 3318-3329, © IEEE 2011. (**Invited Paper**)

4.1 Abstract

Early proponents of public cloud computing have come to identify cost savings a key factor for adoption. However, the adoption and hosting of a web application in the cloud does not provide any such guarantees. This is in part due to the utility pricing model that dictates the cost of public cloud resources. In this work we seek to model and simulate data usage for a web application for the purpose of utility cost analysis. Although much research has been performed in the area of web usage mining, previously proposed models are unable to accurately model web usage profiles for a specific web application nor do they define the necessary analytical metrics needed to measure such accuracy. The first objective of this chapter is to present a simulation model and corresponding algorithm to model web usage based on empirical observations. The validation of the proposed model is performed using four metrics that holistically summarize web usage and results show that the simulated output conforms to that of what was observed and is within acceptable tolerance limits. Building on this work and in the context of the FRC attack, the second objective of this work is to determine the most formidable attack scenario to employ when analyzing the proposed detection and attribution methodologies.

4.2 Introduction

With the advent of the public cloud computing model, web services that were once hosted on private servers and networks are being outsourced to third-party cloud service providers (CSPs) - Amazon's EC2 is a well-known example. Early proponents of this emerging compute model have come to identify cost savings as a key motivation for the adoption of the cloud model. In comparison to more traditional computing models, economic efficiencies in the public cloud have been enabled by the fundamental paradigm shifts in the way computing infrastructure is hosted (e.g., multi-tenant hosting through virtualization, economies of scale, thin provisioning) and the pay-as-you-go business model that dictates costs for resource usages by the cloud consumer (i.e. a person that rents computing infrastructure from a CSP) - namely, the utility compute costing model. However, the adoption and hosting of a web service in the cloud does not provide any guarantees of cost savings as there are many factors that must be taken into consideration [39, 46].

Under the utility compute costing model, much like the utility model that governs the cost of electricity consumption, cloud consumers only pay for the resources they use and only for the time they use them. For instance, the data transfer costs in and out of Amazon's EC2 environment by a cloud consumer's clients (those that patron the cloud-hosted web application) is governed by the Amazon's Web Services costing model (Figure 4.1) and accrues a cost that is a function of the total data transferred [4]. At the conclusion of the month - a typical cloud billing cycle - the aggregated costs are billed to the cloud consumer. Because data usage in the cloud environment is uncertain, the cost for data transfer is as well, which is not typically the case for private web service hosting. The pay-as-you-go billing structure fundamentally changes how those who adopt the cloud model view the monthly data usage of their web applications and motivates the need for modeling and simulation of web usage profiles. Being able to accurately forecast accumulated resource consumption in advance allows one to anticipate costs and manage application designs in order to address costs proactively.

The first objective of this chapter is to model and simulate data usage for a web application for the purpose of utility cost analysis. More specifically, the goal is to explore the minimum

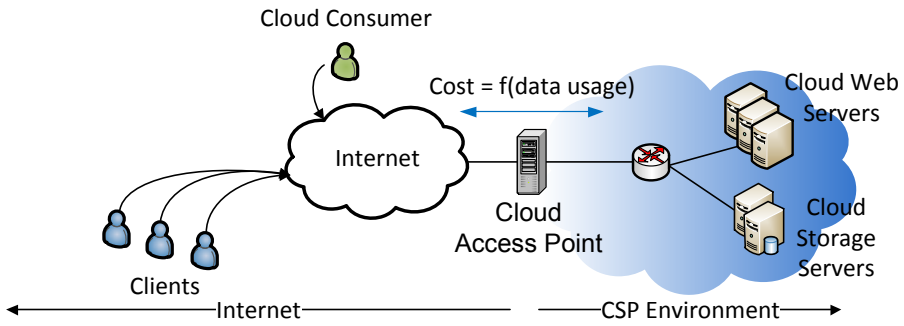


Figure 4.1: Cloud Network Diagram.

number of days of training data necessary to achieve acceptable accuracy of the simulation output. Although much research has been performed in the area of web usage mining - techniques to model and simulate web user transactions - previously proposed models that generate web traffic are unable to accurately model web usage profiles for a specific web application. Either these models generate generic web document requests or such requests are based on theoretical distributions. The approach presented in this chapter differs in that web document requests are derived from trace-driven, and first- and second-order Markov models trained on empirical observations of how the web application under consideration was used in actuality. Additionally, this chapter proposes four analytical metrics with which to measure the holistic accuracy of the simulated output - analysis which is also lacking from previous works. The deliverable is a simulation model and corresponding algorithms to model web usage based on empirical observations. The validation of the proposed model shows that the simulated output conforms to that of what was observed and is within acceptable tolerance limits.

Building on the constructed simulation model for web traffic, this chapter further explores how such a model could be used to generate attack traffic with which to analyze the proposed detection and attribution methodologies. It follows that the second goal of this chapter is to determine which simulation algorithm - trace-driven, first- or second-order Markov - presents the most formidable attack scenario. In other words, which of the listed simulation algorithms would be considered the worst-case scenario for the FRC defender. To make such a distinction, the analytical metrics proposed for the analysis of the simulation model will be applied to a

scenario when a FRC attacker attempts to consume 100% more traffic than is expected.

The rest of the paper is organized as follows. Section 4.3 discusses related works in the context of this work. Section 4.4 describes the dataset used to train and validate the simulation algorithm as well as considerations taken when cleansing the original dataset. To model web usage profiles, in Section 4.5 describes a simulation modeling and corresponding algorithm. Based on this model, Section 4.6 provides the experimental metrics, design, and results used to validate the proposed model. In Section 4.7, this experimental design is used to discover the worst-case FRC attack scenario. Lastly, future work and a conclusion are discussed in Sections 4.8 and 4.9 respectively.

4.3 Related Work

The related works that have bearing on this paper are derived from the research areas of web usage mining and web traffic generation. Although web usage modeling and traffic generation are not mutually exclusive, this chapter will explore a shortcoming in the synthesis of these two research fields. At present, a complete model that takes into account all the necessary sub-models needed to accurately simulate realistic web traffic for a specific website does not appear to exist. Furthermore, there is no model suitable for predictive cost modeling of web traffic. In this section, the described work is briefly describe in the context of these related bodies of work.

Much research has been performed in area of web usage mining since the seminal work done by Arlitt and Williamson [7]. Many of these works, similar to that of [75], [114], and [128] have sought to characterize, model and validate the distributions that depict the way individual users and user populations interact with websites. Extrapolating from this body of research, a number papers have made use of and extended these models to simulate web traffic for a number of purposes. Cao et al. [19] presented a model to simulate generic web traffic on high-speed backbone links. Their objective differs from the objective of this work in that this chapter and the resultant simulation algorithm seeks to model aggregate user behavior for a distinct website as observed by the web server as opposed to modeling link traffic. Luo and Marin [74] devised a model to simulate realistic Internet background traffic, including the web,

for constructing a network intrusion detection environment. Similarly, Kroc et al. [67] focused on modeling the theoretical distributions that together compose a single web user session. While such modeling may be sufficient for background noise and generating realistic user-side web sessions respectively, neither of these two works model specific page requests as observed for a given website. Instead they model web interactions as generic requests. Moreover, the requested web document size is attributed a value based on a theoretical distribution, which is not sufficient for the purposes of accurately modeling actual data usage for a specific website with real document sizes. Instead, specific web requests need to be represented by their known data sizes, which is discussed in Section 4.4.1.

Burklen et al. [17] present a general model and algorithm to synthetically generate a sequence of web requests for a single user. This work is based on known and previously studied web usage behaviors and models in addition to the hyperlink structure of individual web pages and their relationship to other web pages for a given website. While notionally similar to the work in this chapter, the scope of such an algorithm is limited to that of a single user session, not multiple users over a prolonged period of time, which is a key objective of this work. Furthermore, the synthesizing of a single user session is predicated on a theoretical relationship between web pages derived from the site's hyperlink structure and not from leveraging historical observations of how users have traversed the website. We instead generate individual requests that compose a web session from a Markov model based on learned browsing patterns.

In contrast to modeling generic or request sequences based hyperlink structures, Markov chains have been shown to provide accurate models for simulating web usage [71]. Under this guise, Markov models have been used in a number of contexts including performance analysis [21] and caching algorithms [20]. Most similar to the work presented in this chapter are papers that have sought to predict user web sessions by means of Markov models. Nigam and Jain [85] presented a model based on a dynamic nested Markov model for predicting the next page accessed by a user given an observed series of requests. Borges and Levene have produced a number of works - summarized in [10] - that investigate the next page request of individual users and the accuracy of predicting n-grams of requests with various Markov models. While effective for their given purposes, neither of these works provide analysis of

the accuracy and summarization ability of using Markov models for generating and predicting aggregate web traffic based on actual server logs.

In their own respective way, each of these works presented in this section falls short of being able to provide a complete model and simulation framework for data usage transferred by a specific website. To fill this gap, this work creates a synthesis between many of the sub-models presented in these works to provide a more relevant modeling of content usage for a specific website based on training from session logs. This model can then be used for the purposes of modeling usage profiles of cloud-based service for predictive cost analysis and dually for the purposes of FRC attack traffic generation.

4.4 Dataset Description and Considerations

The datasets used for the purposes of this paper are two 56-day web server logs. The first web server log (denoted as *ECpE*) originates from our department's web server and the second log was produced by a busy NASA web server (denoted as *NASA*) [82]. To demonstrate the generality of the proposed algorithm, each of the web logs are used to both train the proposed model as well as for the experimental validation of the simulation proposed algorithms presented in Section 4.5.

Web server usage is often represented as Zipf-like distributions [14] in which the frequency of a requested document $p(i)$ is proportional to its rank i such that $p(i) \propto 1/i^{\alpha}$ where α is close to unity. Figure 4.2 depicts a Zipf-like distribution for the *ECpE* dataset as a log-log plot of request frequency vs. rank. Figure 4.3 shows a similar log-log plot of document rank vs. data usage. Drawing from Figure 4.2, the 356 most requested pages represent 90% of all requests and of these pages their weight in data usage totals over 97% of data requested (Figure 4.3) over the observed 56-day span for the *ECpE* dataset. Given these empirical distributions and observations, the modeling of data usage for the given dataset is heavily dependent upon accurately modeling the most frequently requested documents.

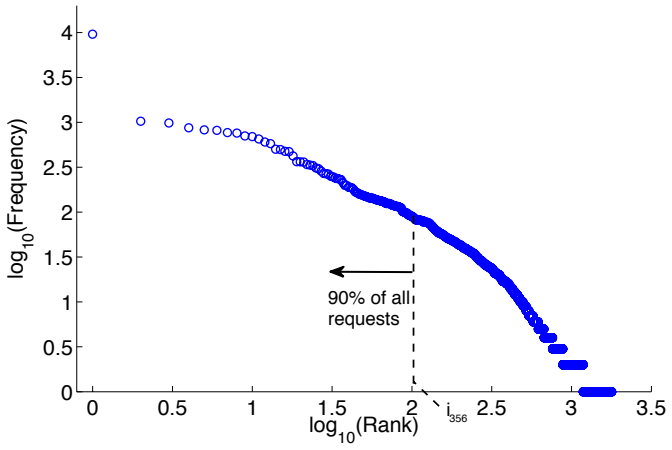


Figure 4.2: Zipf-like Distribution for Request Frequency.

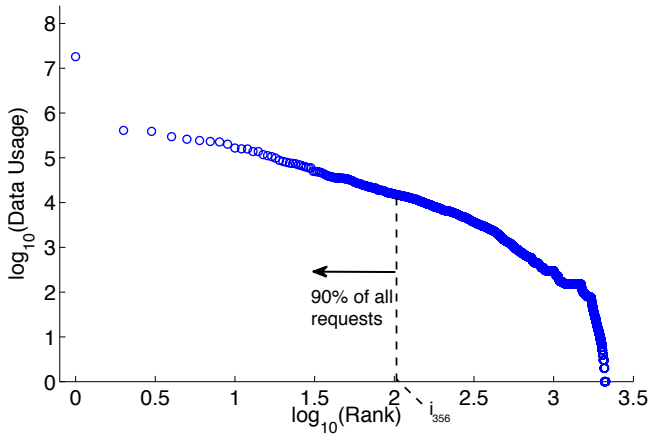


Figure 4.3: Zipf-like Distribution for Data Usage.

4.4.1 Web Usage Mining and Modeling Components

A web server log maintains an itemized journal of all users' content requests and provides the necessary observations for deriving empirical distributions and models used in the study of web usage. Figure 4.4 provides an illustration of web usage metrics and will be used as a guide to explain these metrics as well as the considerations taken to cleanse the observed web server log from its original form to what was used for the purposes of this chapter.

Primary Web Document Request - An individual web request within a web log is depicted as a vertical line in Figure 4.4. As seen annotated in the call-out, requests are composed of an IP address for the requesting client, time stamp, document requested, and the data size

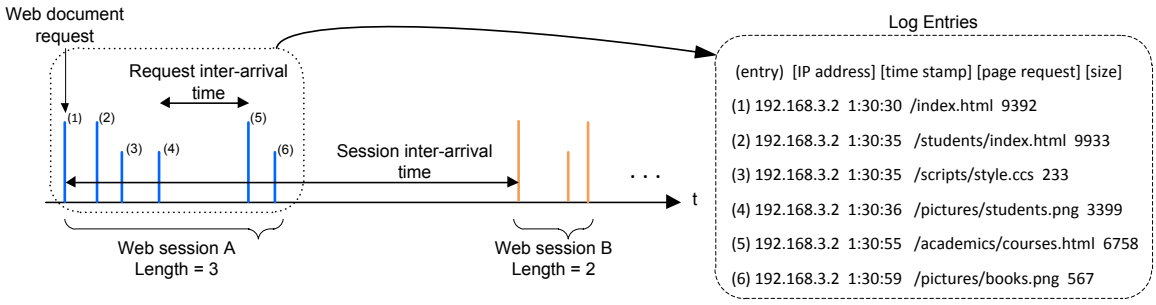


Figure 4.4: Web Usage Modeling Components.

of the respective document.

The modeling and simulation objectives of this work are reliant on the accuracy of the data size attributed to each client-invoked request (i.e. a primary request). The data size, as shown in Figure 4.4, for each individual HTML request can be misleading as it is not a complete account of the data usage needed to view the expected web page, but instead the entry only reports the size of the HTML file itself. Typically, a single request for a HTML document invokes other secondary in-line requests to retrieve embedded objects within the primary HTML page such as pictures, scripts, and videos. In Figure 4.4, both (3)/*scripts*/*style.css* and (4)/*pictures*/*students.png* are secondary in-line requests, shown as shorter lines, of the primary request, shown as a taller line, for (2)/*students*/*index.html*. Together, the size of the primary and associated secondary requests represent the total data usage for a single client-invoked request, which is one of the objectives of this analysis. While both primary and secondary requests are registered in a web log, archival analysis is inadequate to determine whether an entry is a primary or a secondary request, and relate secondary requests with its parent primary request. Client-side and distributed caching further complicate the task of reconstructing these relationships from a web log.

Therefore, in lieu of these impediments and in order to accurately capture the data size for each primary client-invoked request and its accompanying secondary requests, analysis was performed on the active departmental website (i.e. the *ECpE* dataset). The URL for each primary request in the web log - assumed to be an HTML document or URI - was requested from the website with a script capable of capturing the data usage footprint for the primary request as well as all secondary requests. After accumulating the footprint of each primary

request and relevant secondary requests, the total replaced the original primary request size and the secondary entries were discarded. Such analysis and post-processing can only be performed with access to a live website and thus such analysis was not performed on that *NASA* dataset. Therefore, the presented initial results for data usage modeling are limited to that of a single website, since access to web logs of an active web site is severely limited.

Web Session - A web session is a set of consecutive requests generated by an individual user during a single viewing period. As seen in Figure 4.4, *web session A* contains three primary web requests and thus has a web session length of three. A web log is composed of many interleaved web sessions initiated by multiple users. Within the observed web logs described in this section, web session lengths ranged from well over 100 documents in length with some sessions as long as 1400 primary requests. In order to provide a more accurate modeling of how the majority of normal users actually traversed the website, web session lengths were truncated to 35 primary requests, which falls within the 99th percentile.

Often web logs do not contain the complete information necessary to discern when a web session for a user ends and when the user's next web session begins. Research in this area has sought to differentiate between sessions using time-oriented heuristics [130] and transitional request probabilities using a first-order Markov model [92]. However, for simplicity, it is assumed in this work that a 900 second or greater time lapse between primary requests denotes the end of one web session and the beginning of a new session. This assumption is consistent with previous works in the field [67].

Session Inter-arrival Time - Session inter-arrival time is the measure of time between the beginning of two consecutive web sessions. In Figure 4.4, the session inter-arrival time is depicted as the time between the beginning of *web session A* and that of *web session B*.

Request Inter-arrival Time - Within a given session greater in length than one, there is an intermittent amount of time experienced between each respective primary web request by the client. This is referred to as the request inter-arrival time and is shown between *web session A* requests *(4)* and *(5)* in Figure 4.4.

4.4.2 Dataset Limitations

As with many empirically based models, the simulation results are heavily dependent on the quality of the training data. Moreover, the training of trace-driven and Markov models based on the actions that have been observed in the web logs restricts the model to only the web pages requested and conditional probabilities between pages that have been observed. Lastly, due to the necessity of performing analysis on a live website and limited cooperation of website operators, the results of this work is currently limited. Although there are no indications that the presented model and algorithm would not provide a general solution, such claims can only be made after further analysis of a broader set of websites.

4.5 Simulation Algorithms

The objective of the proposed simulation algorithms are to generate web traffic in accordance with what has been observed. The uniqueness of the outlined approaches in comparison to the papers discussed in Section 4.3 is that the described simulation algorithms generate web traffic crafted from empirical distributions from a specific web application and utilizes trained models to generate page requests that reflect actual primary request patterns. In this section, modeling considerations and algorithms based on first-order and second-order Markov models in addition to a trace-drive simulation model will be discussed. The primary objective of this section is to describe a proposed Markov-based simulation algorithm by dissecting a second-order Markov model and its underlying modeling components. Although not overtly described in the same detail as the algorithm for the second-order Markov model, a similar algorithm using a first-order Markov model was also constructed for modeling web usage.

4.5.1 Description of Markov-Based Simulation Algorithms

Given a web server log composed of N days of observed requests as an input, the objective of Algorithm 1 is to simulate a web server log that conforms to the empirical distributions derived from the input dataset while preserving web usage behaviors as they were deduced from sessions within the input web log. The output of the simulation algorithm is a web server

log L composed of many users' web sessions that emulates actual clients as they utilize the website over a period of time.

A web server log L is composed of many independent and, at times, overlapping sessions s that represent the actions taken by a website's user base. Each session $s \in L$ is a tuple $s =< ipAddress, sessionLength, P >$ that is composed of the IP address of the individual requester, the number of web pages requested during a given session, and the set of primary web page requests P. Each $p \in P$ is also a tuple $p =< page, time, size >$ that denotes the specific web page, time stamp, and size of each individual web page request within a session.

Algorithm 1 Modeling Web Usage from a Second-Order Markov Model

Require: Observed Web Server Log
Ensure: Generated Web Server Log

```
1: generateLog() : L {
2: absoluteTime ← 0, i ← 0, currentRequests ← 0;
3: while currentRequests < totalRequests do
4:    s_i.sessionLength ← generateSessionLength();
5:    s_i.ipAddress ← generateIpAddress();
6:    absolute_time += generateSessionInterarrivalTime();
7:    currentRequests+ = s_i.sessionLength;
8:    for j ← 1 to s_i.sessionLength do
9:       if j == 1 then
10:         p_j.page ← returnFirstPage();
11:         p_j.time ← absoluteTime;
12:         relative_time ← absoluteTime;
13:      else
14:         if j == 2 then
15:            p_j.page ← returnPageFirstOrderMarkov(p_{j−1}.page);
16:         else
17:            p_j.page ← returnPageSecondOrderMarkov(p_{j−1}.page, p_{j−2}.page);
18:         end if
19:         relativeTime += generateRequestInterarrivalTime();
20:         p_j.time ← relativeTime;
21:      end if
22:      p_j.size ← pageSize();
23:   end for
24:   L = L ∪ s_i;
25:   i++;
26: end while
27: return L;
28: }
```

4.5.2 Algorithm Modeling Components

The presented algorithm illustrates a high-level overview of this approach to simulating web usage profiles. The underpinnings of the algorithm are derived from published web usage metrics in coordination with generating primary request sequences by using a Markov model. The following descriptions provide a thorough analysis of the algorithm components used for experimental evaluation, which is reviewed in the next section.

Line 3: *totalRequests* - Although not formally presented in in this paper, linear regression analysis was performed on each training dataset to extrapolate the expectation of the number requests for the given target of accumulated simulation days. This value was used as the control parameter to dictate the length of each simulation run.

Line 4: *generateSessionLength*() - The session length defines the number of primary web requests by an individual user during a single browsing period. Session lengths were modeled as a Lognormal distribution with the following parameters: α = 0.44-0.48, β = 0.76-0.78, μ = 2.47-2.49 pages, σ = 3.45-3.46 pages. The modeled session length distribution is consistent with [114].

Line 5: *generateIpAddress*() - The IP addresses chosen for each individual session were modeled and drawn from a continuous, piecewise-linear empirical distribution that was populated based on the pre-processing analysis of the training data set.

Line 6: *generateSessionInterarrivalTime*() - The session inter-arrival times were modeled as an exponential distribution with a mean that varied between 113 and 126 seconds. Although in [67], session length was modeled with a Weibull distribution, we found an exponential distribution to be a more appropriate fit for the given data sets.

Line 10: *returnFirstPage*() - Each simulated session is initialized by determining the first page to be synthetically generated for a given user and for each individual session. The first page distribution is an initial state vector learned from the first page views of the sessions extracted from the training dataset. Due to the self-similarity of web traffic, the distribution of first page requests can also be depicted with a Zipf-like distribution

similar to that of the aggregate request distributions presented in Section 4.4. Figure 4.5 represents the first-page distribution of the training data set and Figure 4.6 represents the simulated first-page distribution.

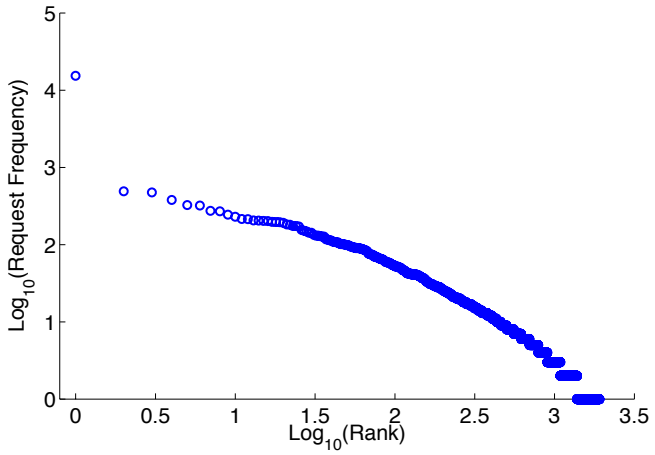


Figure 4.5: Actual First-Page Zipf Distribution.

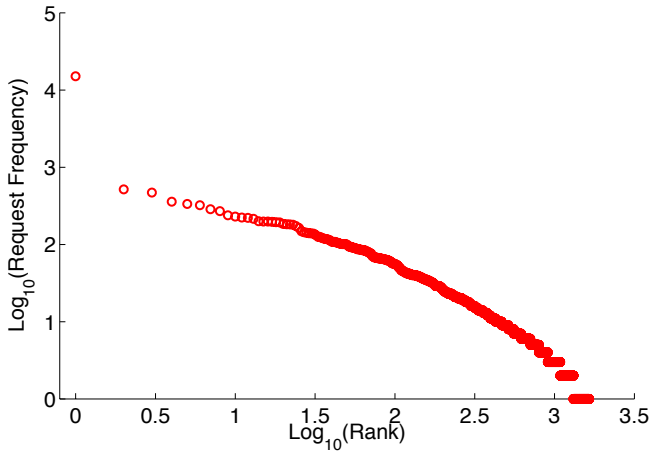


Figure 4.6: Generated First-Page Zipf Distribution.

Line 15: *returnPageFirstOrderMarkov*() - A Markov model is used to generate the actual page requests and is trained by analyzing web server logs. Based on the presented algorithm, if a session length is at least two, the second request generated in a web session is drawn from a first-order Markov model such that $p_{ij} = Pr(x_{n+1} = j | x_n = i)$. For a thorough explanation of training and building Markov models, which is accompanied examples, please see [10].

Line 17: $returnPageSecondOrderMarkov()$ - For web session lengths greater than two, all subsequent requests are generated from a second-order Markov model such that $p_{ijk} = Pr(x_{n+2} = k|x_{n+1} = j, x_n = i)$. The second-order model was constructed in a manner similar to the first-order model.

Line 19: $generateRequestInterarrivalTime()$ - Each request inter-arrival time was generated from a Weibull distribution with the scale parameter ranging between 28.57 and 33.8 and the shape parameter between 0.57 and 0.62. The fitting of this distribution aligns with that in [67].

4.5.3 Description of Trace-Driven Simulation Algorithm

Different than a Markov-based model, a trace-driven simulation model is simply the replaying of randomly chosen web sessions as they were experienced in the training dataset. Instead of constructing conditional probabilities between successive requests, in this model, the selection of primary web requests is a product of a randomly chosen web session that reflect exactly how real users traversed the website under consideration. To construct such a model, web sessions were extracted from the training dataset and assembled into datastore that allowed for both random selection and quick retrieval. Unlike Algorithm 1, the $generateSessionLength()$ function for the trace-driven algorithm was not determined by a theoretical distribution but instead by the empirical distribution of actual sessions and their respective lengths. Furthermore, because of the prescribed nature of this model, the $returnFirstPage()$, $returnPageFirstOrderMarkov()$, and $returnPageSecondOrderMarkov()$ functions were replaced by a much simpler function that injected the appropriate primary web request from the chosen web session selection. To preserve the flexibility of the simulation and modeling ability of the Markov-based algorithm, the simulation of IP addresses and inter-session and inter-arrival times is similarly performed for the trace-driven algorithm. The hypothesis tested by introducing the trace-driven algorithm is whether or not first-order or second-order Markov models are better equipped at modeling web usage in comparison to replaying what was observed.

4.6 Experimental Evaluation

This section presents the experimental metrics, experimental design and related experimental results. Validation of the described algorithms relies on four key measures : 1) relative proportionality and rank of aggregate requests (Zipf Metric - Section 4.6.1.1) 2) summarization accuracy (Spearman Footrule Distance - Section 4.6.1.2), 3) summarization completeness (Overlap Metric - Section 4.6.1.3), and 4) modeling of data usage error (Section 4.6.1.4). Together these four metrics are used to assess the output of the simulation models relative to actual web usage recorded by the respective websites.

4.6.1 Experimental Metrics

4.6.1.1 Zipf Metric

As described in Section 4.4, a Zipf-like distribution for a web dataset can be depicted by constructing a log-log plot of requested document frequency vs. rank. Through experimentation of Zipf-like distributions of web datasets, it has been found that constructing a linear regression line of the top 10% of primary requests yields a consistent metric with which to measure the relative proportionality and rank of aggregate web documents. The capturing of only the top 10% of web documents is motivated by the inconsistencies in the tail of Zipf-like distributions formed from web logs. Figures 4.7 and 4.8 represent Zipf values for both an actual and simulated web datasets.

Because the Zipf metric is an empirical measure of a single dataset, the percent error (Equation 4.1) between between the actual Zipf value for the target web log and the corresponding Zipf value (i.e. the slope of the regression line) for the simulated results were calculated to produce a consistent metric with which to compare and analyze.

$$PercentError = \frac{Experimental - Actual}{Actual} * 100 \tag{4.1}$$

Due to the fact that the Zipf metric only measures the relative volume of aggregate requests between the expected and simulation web logs, it does little to express the actual accuracy and

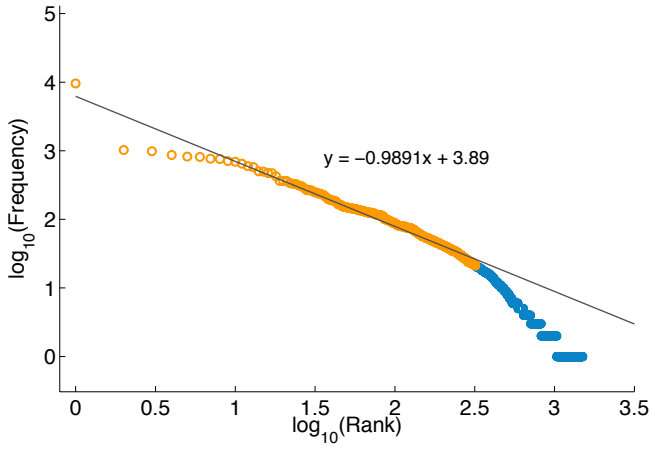


Figure 4.7: Actual Zipf-like Distribution

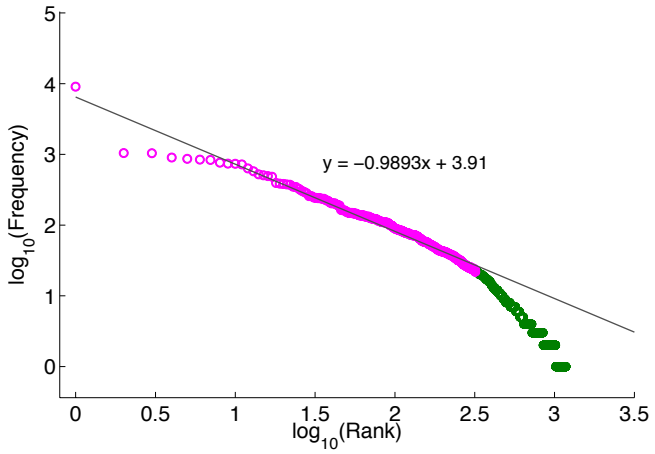


Figure 4.8: Simulated Zipf-like Distribution

completeness of the web documents that were requested. For this analysis, the Spearman Footrule distance and the overlap metrics are introduced next.

4.6.1.2 Spearman Footrule Distance

The Spearman's Footrule distance [32] is a non-parametric measure of association between two ranked lists. This measure was used in a similar context to measure the accuracy of predicting individual session n-grams generated from Markov models [10]. For this work, however, the Spearman's Footrule distance is instead utilized to measure the summarization accuracy of the simulation algorithm by analyzing the ranked lists of the top-10% of the simulated primary

requests output in comparison to that of the web logs.

The purpose of employing the Spearman's Footrule is to find an aggregated ranking that minimizes the distance between two ranked lists. However, for the purposes of this paper, the proximity between two ranked lists will instead be considered as it is a more appropriate measure that aligns with the described analysis objectives.

Drawing on the notation established in [10], the Spearman's Footrule proximity is defined as follows: Given two ranked top-k lists L_1 and L_2 as inputs, with each list containing k entries, let L be the union of the two lists such that $L = L_1 \cup L_2$. Furthermore, let L_1 be the reference list that is assumed to be the ground truth and L_2 be the comparison list, which in all actuality is a partial list in comparison to that of the reference list. To obtain the ranking of a list item $i \in L$ in L_1, we define the function $f(i)$ and similarly $g(i)$ for $i \in L_2$. In either function $f(i)$ or $g(i)$ if $i \notin L$, then the subsequent ranking is assigned that of a location parameter $l = k + 1$ [105]. Given these preliminaries, the Spearman's Footrule proximity is defined as follows:

$$F(L_1, L_2) = 1 - \frac{\sum_{i \in L} |f(i) - g(i)|}{k(k+1)} \tag{4.2}$$

In order to provide a measure of similarity or proximity instead of a measure of difference or distance, the normalized summation in Equation 5.3 is subtracted from one. In the case that both ranked lists were identical, the Spearman proximity would be one. It follows that two disjoint lists would yield a Spearman proximity value of zero.

To provide a tangible example for the calculation and purpose of the Spearman proximity measure, Table 4.1 provides a modified listing of the top-15 requested web pages for both the *ECpE* dataset (i.e. Actual) - reference list - and the simulated results using first-order Markov model as described in Section 4.5 (i.e. Simulated). The respective *Spear* values, according to Equation 5.3, can be found in the fourth column in Table 4.1. As can bee seen, the 10th most popular web age for the Actual dataset is not present in the Simulated dataset, which incurs the extraction penalty of 16 (i.e. k+1). Once summed and normalized, the provided example results in an overall Spearman proximity value of 0.73.

Table 4.1: Spearman and Overlap Comparison

Rank	Reference List (L_1)	(L_2)	Spear
1	/	/	0
2	/who-we-are/faculty-new.html	/who-we-are.html	1
3	/who-we-are.html	/who-we-are/faculty-new.html	1
4	/academics/courses.html	/academics/courses.html	0
5	/research.html	/students/graduate-students.html	3
6	/students/graduate-students.html	/students.html	1
7	/students.html	/academics.html	1
8	/academics.html	/research.html	1
9	/admissions/grad-guidelines.html	/admissions/grad-guidelines.html	0
10	/academics/calendar.html	/who-we-are/staff.html	16
11	/admissions.html	/academics/ee-major.html	1
12	/learning.html	/admissions.html	16
13	/academics/flowcharts.html	/academics/flowcharts.html	0
14	/research/funding.html	/research/research-groups.html	16
15	/who-we-are/people.html	/academics/cpre-major.html	16
Spearman	**0.70**		

4.6.1.3 Overlap Metric

In conjunction with the Spearman's Footrule proximity, the overlap between the reference list L_1 and the comparator list L_2 is measured to provide a broad indication of the summarization ability of the simulation model output. The overlap is defined as the percentage of items in the comparator list L_2 that appear in the reference list L_1 such that:

$$O(L_1, L_2) = \frac{|L_1 \cap L_2|}{|L_1|} \tag{4.3}$$

In comparison to the Spearman distance, the overlap value measures the completeness between the two analyzed datasets. As seen in Table 4.1, a similar example is provided to demonstrate the calculation for the overlap measure. From this analysis, it is shown that the Simulated dataset contains 11 of the 15 web documents contained in the Actual dataset (i.e. the reference list). As a result, the overlap value for the two lists is $11/15 = 73\%$.

4.6.1.4 Percent Error

Lastly, the percent error (Equation 4.1) between the expected value of the aggregate data usage (in bytes) as produced by the output from the simulation model and the actual data usage from that of the observed logs is measured. This measure compliments the Zipf, Spearman and overlap values to provide a comparative indication of the model's ability to accurately forecast

aggregate data usage, which is a key variable in public cloud utility costing models.

4.6.2 Experimental Design

The available datasets were utilized to there full extent by using a sliding window for both the input training days and for the output comparison. A goal of this work was to explore the minimum number of training days necessary to accurately simulate future aggregate use. In the context of cloud computing and due to monthly billing cycles, future aggregate use is most appropriately defined as the 30 days in advance of the first observed day of the given training window. For example, as illustrated in Figure 4.9, the *simulation run A* is trained on the days 1-10 of accumulated logs and is tasked with simulating the aggregate usage of the web application from days 1-30. The results of such a simulation output are then compared to that of the observed web log. To provide multiple simulation runs for a given training window size, the simulation was repeated by shifting the simulation window into the future one day and repeating. As shown in Figure 4.9, the *simulation run B* was trained on the logs from days 2-11 and was tasked with simulating the aggregate usage for days 2-31 and thus a sliding training window of 10 days.

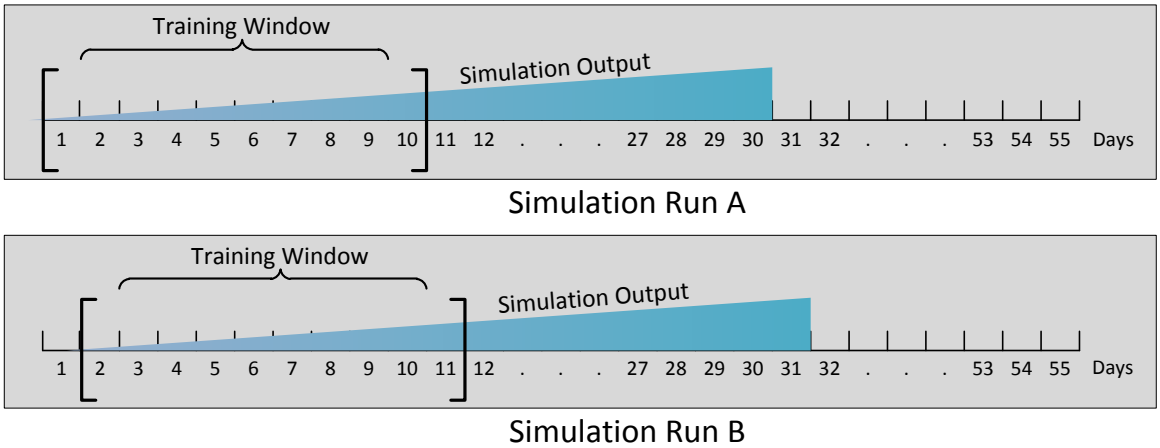


Figure 4.9: Experiment Simulation Design.

To explore the minimum number of training days necessary to simulate future aggregate use, a sliding window of observed days ranging from 4 days to 29 days was used. The size of the training window (in days) is denoted as the x-axis for the Figures in both Section 4.6.3

and 4.7. For each training window size, 25 simulation runs were conducted and the metrics described in Section 4.6.1 were calculated. Each datapoint on the following figures represents an average for the given metric resulting from 25 simulations runs per training window size.

4.6.3 Experimental Results

To generate web sessions that together form a web log, experimental simulations were performed using trace-driven and both first- and second-order Markov models. Having prior access to the accumulated data logs allowed for the comparison of the simulation model output with that of what actually transpired.

Figure 4.10 provides a comparison the of percent error in accumulated data usage between generating web requests from a trace-drive and first- and second-order Markov models. From the analysis of the $ECpE$ dataset, it can be seen that the trace-drive simulation model provides a more accurate modeling of data usage than both of the Markov-based models for training window sizes larger than four days. While it takes approximately eight days of observed web logs for the trace-driven model to produce a 30-day projection of data usage that is within a 5% error tolerance of the actual value, it takes more than twice as many observed days to for the Markov-based models to achieve the same results. As expected, the initial prediction capability of the three models improves as the training window sizes increases. However, the trace-drive model reaches a limitation of accuracy after approximately 21 days while the first- and second-order Markov models exhibits a more linear improvement in accuracy after seven days while maintaining a proportional relationship as the training window size increases. While similar analysis of the $NASA$ dataset would be beneficial, the historic nature of the website did not lend itself to being able to accurately attribute secondary-line inline requests to their respective primary requests and thus it was not possible to accurately model the total data footprint for each primary request. Despite this drawback, the remaining metrics previously discussed are applicable to both the $ECpE$ and $NASA$ dataset.

Figures 4.11 and 4.12 show the Spearman's Footrule proximity between the simulated output and the observed logs for the top-10% of requests for both the $ECpE$ and the $NASA$ datasets. The results show that across all training window sizes, the trace-driven model provides a

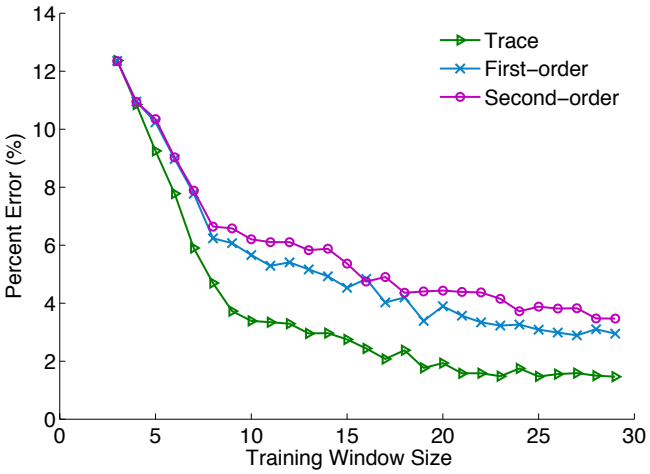


Figure 4.10: ECpE - Data Percent Error.

consistently better *accuracy* in summarization ability in comparison to that of the Markov-based models. This is in part due to the nature of Markov models. In this context, first-order models accurately represent the first two requests of a session but do not accurately represent all second-order conditional probabilities for session lengths greater than two. Second-order models on the other hand, accurately model second-order conditional probabilities and thus have a higher accuracy in reflecting reality but do so at the loss of coverage. The completeness of the summarization strengths of the trace-driven can further be seen in Figures 4.13 and 4.14, which provides a higher-level comparison of the overlap, or the *completeness* in summarization between the top-10% of requests.

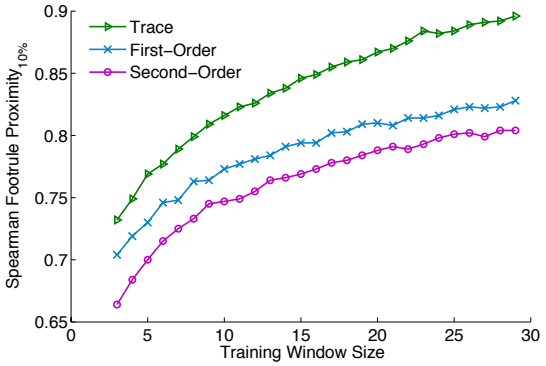


Figure 4.11: ECpE: Spearman's Proximity.

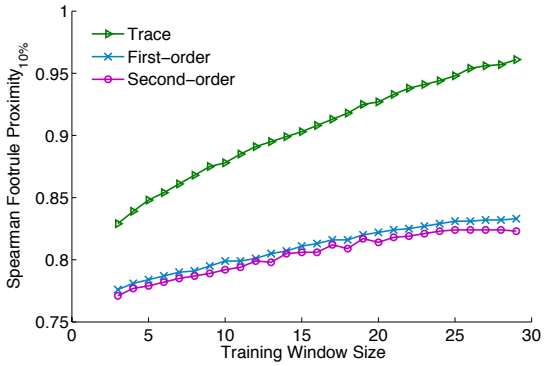


Figure 4.12: NASA: Spearman's Proximity.

Similar to the Spearman proximity analysis, the overlap between the top-10% of requested

documents yields results that show the trace-driven model is more adept at providing a more accurate completeness of summarization between the three analyzed simulation models.

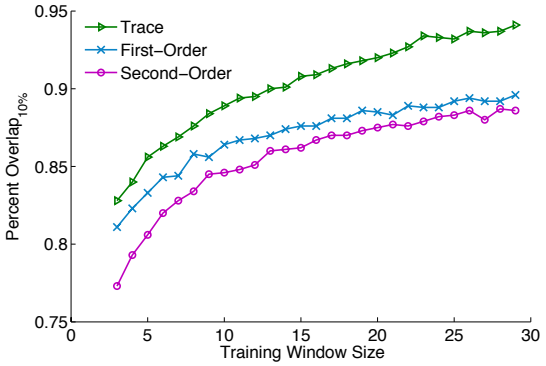


Figure 4.13: ECpE: Overlap Percentage.

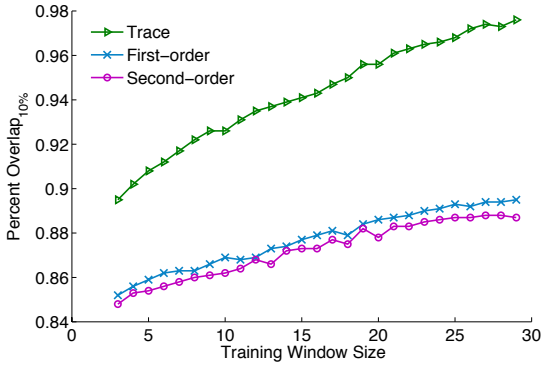


Figure 4.14: NASA: Overlap Percentage.

To complement the data usage and summarization (i.e. Spearman and overlap) metrics, the Zipf analysis provides a measure of the relative proportionality of request volume between the top-10% of requested documents. Because the aforementioned summarization metrics are non-parametric measures, they do not account of the volume of requests attributed to each respective primary request. As can been seen in Figures 4.15 and 4.16, the Zipf value for the trace-driven model provides a significantly better result in comparison Markov-based models, which is consistent the previously analyzed results. These results are also a product of the decreased coverage of Markov models and the bias they create towards the most popular documents.

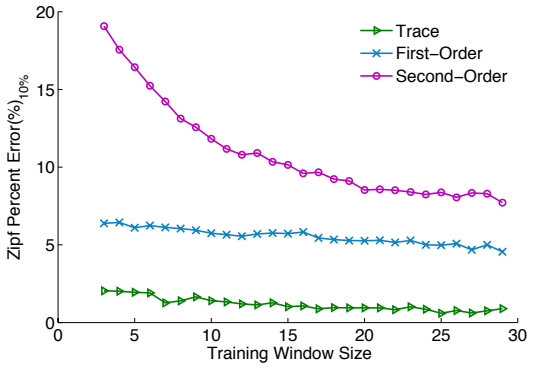


Figure 4.15: ECpE: Zipf Value.

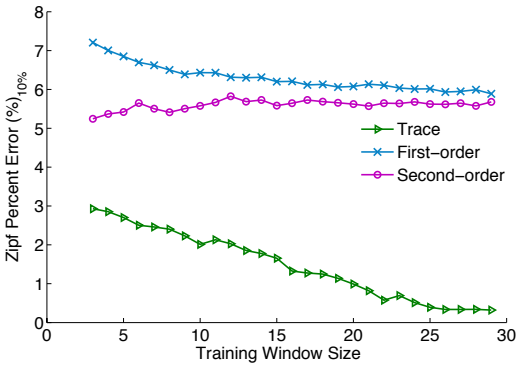


Figure 4.16: NASA: Zipf Value.

Through this study, a trace-driven simulation model has been shown to provide a consistently more accurate modeling of web data usage than a either a first- or second-order Markov

model. In addition to the less accurate ability to model web usage, the drawback of Markov models, in general and as they increase in order, is that they consume a exponentially higher state-space and longer model runtimes than that of lower-order or trace-driven models. For the application of data usage forecasting a trace-driven model has been shown to be definitely the best of the analyzed models in all four provided measures.

4.7 Attack Traffic Generation

As discussed in the Section 4.2, the provided algorithm and results serve a dual purpose for this work. Although the primary focus of this chapter was centered on modeling and simulating web traffic for a specific website or web application, the presented results can also be used to generate realistic attack traffic that can be used for the purposes of testing proposed FRC detection and attribution solutions. Formal modeling enables for the creation of worst-case attack scenarios that can test methodologies under challenging scenarios.

Different than the analysis in Section 4.6.3, the objective of this section is to simulate web usages in volumes that would be indicative of a FRC attack. To test the three simulation models adeptness at creating realistic attack data in excess of normal, the experiments provided in this section were constructed to consume 100% more requests than the previously targeted 30-day value. Comparing the actual 30-day value from the observed web logs with the simulated attack data will provide the answer as to what simulation methodology would best be suited for the purposes of simulating a stealthy FRC attack.

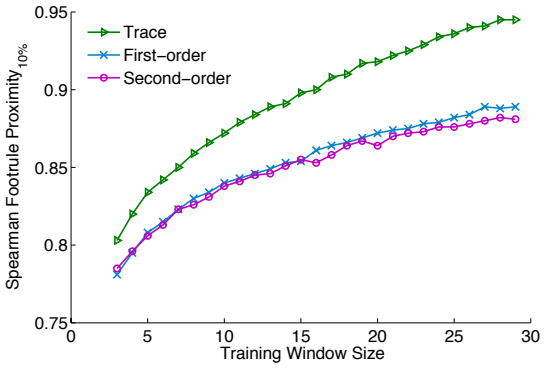


Figure 4.17: ECpE: Spearman's Proximity.

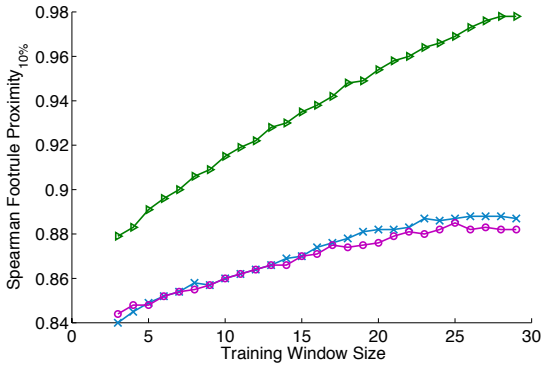


Figure 4.18: NASA: Spearman's Proximity.

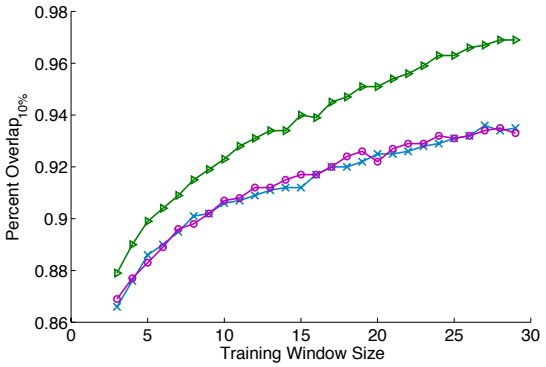


Figure 4.19: ECpE: Overlap Percentage.

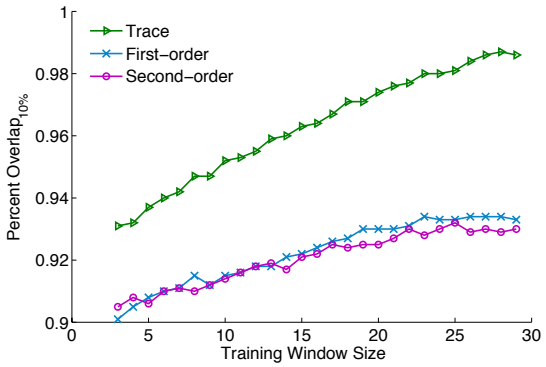


Figure 4.20: NASA: Overlap Percentage.

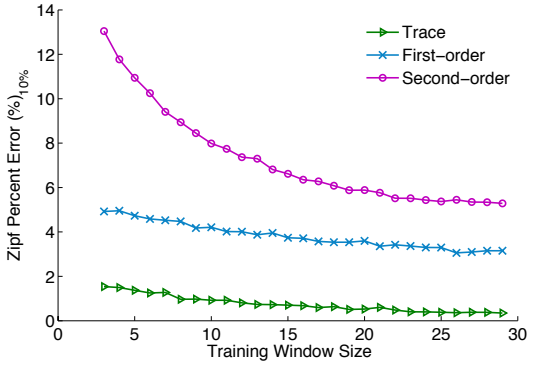


Figure 4.21: ECpE: Zipf Value.

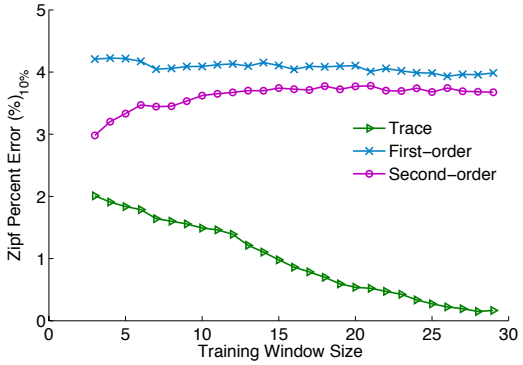


Figure 4.22: NASA: Zipf Value.

Similar to the experimental results presented in Section 4.6.3 and despite the 100% increase in simulated traffic volume, the trace-driven model outperformed both the first- and second-order Markov models in respect to the Zipf, Spearman, and overlap metrics. As a result, throughout the rest of this work, the trace-driven attack scenario will be considered the most formidable attack.

4.8 Future Work

Further validation and future work on this topic will be heavily dependent on obtaining a sufficient number of daily web logs from an active websites of sufficient length and that are suited for data usage modeling. To test the generality of the result presented, it would also be beneficial to obtain numerous web logs from diverse web sites or web applications and of varying user volume. While desirable, privacy and security concerns are significant impediments

to accessing web logs for research use. In order to make due with what is available, and thus to render existing publicly available datasets useful for such analysis, heuristic-based approaches will be necessary in order to identify primary requests and to reconstruct data usage footprints from associated secondary in-line requests. Within the study of Markov models, there exist a number of methodologies that could be implemented (or expanded upon) to further refine the accuracy of session generation components of the model, which could potentially lead to more precise data usage modeling and provide for a more accurate coverage of overall requests. In addition to the modeling of web usage for utility cost analysis, many other aspects of a CSP platform like server instant hours, data I/O, and storage could be modeled and simulated to provide a more holistic account the total expenses a cloud consumer incurs.

4.9 Conclusion

Resource planning is not unique to where an application is hosted. However, having accurate profiles of future web application usage allows for more efficient management and expectations of costs in the cloud. In order to address this challenge, modeling needs to be able to characterize a given web application trained with actuals usage patterns. In summary, the presented approach was to do the following: 1) obtain actual page sizes (i.e. size of primary and secondary in-line requests); 2) determine empirical distributions from actual web logs; 3) build Markov models that represent page request order based on actual usage; 4) apply an algorithm that leverages the models in 2) and 3) to generate web log entries for the target number of days; and 5) evaluate accuracy and summarization of the resultant logs compared with actual logs. In a practical setting the last step would not be possible until after the target day has passed, but step 5) would be helpful to establish ongoing confidence in and possibly tune parameters within the approach.

This paper contributes to the field of modeling and simulation by offering a modeling approach that approximates actual web application usage behavior with greater fidelity by tailoring modeling to the application instance. Moreover, the developed algorithm provides a means to utilize these models to produce days of complete web server logs. Thus providing one an ability to evaluate the qualities of this approach with a practical benchmark. Results

have shown that a minimum of eight days of observed logs coupled with a trace-driven model is necessary to provide a sufficiently accurate projection of logs for a cloud billing cycle. Lastly, it was shown that the trace-driven model was best suited to provide the most formidable FRC attack scenario.

CHAPTER 5. FRC DETECTION

Chapter contains modified content from the following published conference paper:

Idziorek, J., Tannian, M. and Jacobson, D. "Detecting Fraudulent Use of Cloud Resources." *In Proceedings of the 2011 ACM Workshop on Cloud Computing Security (CCSW '11) at CCS.* Chicago, IL. 21 Oct. 2011. pp. 61-72, © 2011, Association for Computing Machinery, Inc. (**Acceptance Rate 13/45 = 29%**)

5.1 Abstract

Initial threat modeling and security research on the public cloud model has primarily focused on the confidentiality and integrity of data transferred, processed, and stored in the cloud. Little attention has been paid to the external threat sources that have the capability to affect the financial viability, hence the long-term availability, of services hosted in the public cloud. Similar to an application-layer DDoS attack, a Fraudulent Resource Consumption (FRC) attack is a much more subtle attack carried out over a longer duration of time. The objective of the attacker is to exploit the utility pricing model which governs the resource usage in the cloud model by fraudulently consuming web content with the purpose of depriving the victim of their long-term economic availability of hosting publicly accessible web content in the cloud. This chapter thoroughly describes the FRC attack and discusses why current application-layer DDoS detection schemes are not applicable to a more subtle attack. The chapter goes on to propose three detection metrics that together form the criteria for identifying a FRC attack from that of normal web activity. Experimental results based on three plausible attack scenarios show

that an attacker without knowledge of the training web log has a difficult time mimicking the self-similar and consistent request semantics of normal web activity.

5.2 Introduction

The inevitable trade-off for the potential cost savings and convenience afforded by public cloud computing is the exposure to both new and old security risks. While both the confidentiality and integrity of data transferred, processed, and stored in the cloud has attracted much of the focus from initial threat modeling and research, little attention has been paid to the external threat sources that have the capability to affect the financial viability and thus the long-term availability of services hosted in the public cloud.

A defining characteristic of the public cloud model is the pay-as-you-go pricing model [77] that governs resource consumption costs (e.g. server hours, bandwidth, storage, etc.) - more broadly known as utility pricing. Under this pricing structure, public-facing web content is vulnerable to remote exploitation in which attack clients purposefully request web content in volumes that are economically unsustainable for the cloud consumer - one that rents services from a Cloud Service Provider (CSP) (Figure 5.1).

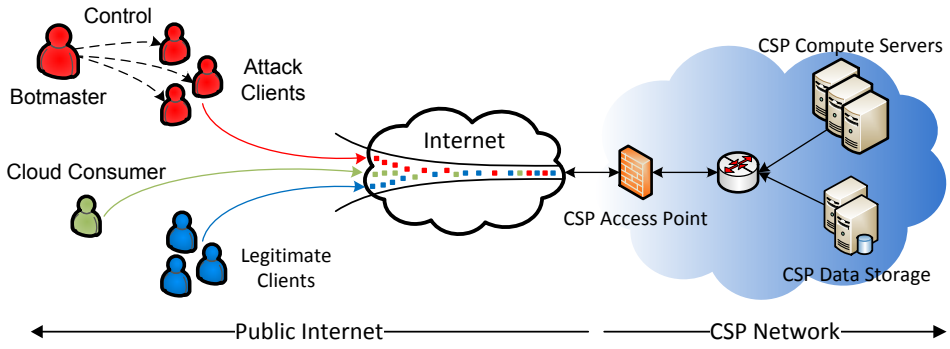


Figure 5.1: Cloud Network Attack Diagram

Notionally similar to an application-layer DDoS attack, a Fraudulent Resource Consumption (FRC) attack is a much more subtle attack and is carried out over a longer duration of time. The objective of the attacker is to exploit the utility pricing model by fraudulently consuming web content with the purpose of depriving the victim of their long-term economic viability of hosting

publicly accessible web content in the cloud. To avoid detection by current application-layer DDoS solutions, fraudulent request rates of attack clients are of moderate intensity and requests attempt to blend into the normal activity of the target website. For the cloud consumer, differentiating data usage of legitimate clients from that of attack clients is difficult because requests only differ in the intentions of the attacker not in the structure or semantics of the requests.

Known as the *free-rider problem* in field of economics [59], the unobstructed access to web resources (i.e. non-excludable goods) by the general nondescript public via the Internet creates a scenario in which the cost of excessive resource consumption is shouldered not by the requesting users but instead by the cloud consumer. In this context, excessive consumption leads to market failure for the victim and abandonment of the public cloud model. The exploitation of utility models is not a new concept. Telephone networks have experienced their fair share of similar fraudulent consumption in the past [36]. However, unlike the telephone networks, which have since abandoned in-band signaling and thereby addressing a well-known vulnerability, public access and the pay-as-you-go business model are an architectural dichotomy that remain at the root of this problem. Until the systemic vulnerability is addressed, detection and mitigation of fraudulent usage of the public cloud should commence in earnest. At this time, there appear to be no viable detection mechanisms capable of differentiating a FRC attack from that of a normal increase in the web traffic volume for a given website. This work seeks to contribute to the effort of addressing this detection gap.

In the context of this chapter, the term *detection* is defined to be a method of distinguishing a FRC attack from that of normal user activity and *attribution* to be the process of properly identifying individual attack clients. The focus of this chapter is to provide a thorough analysis of FRC attack detection methodologies. Attribution of such attacks is reserved for Chapter 6. The contributions of this paper are threefold. (1) The FRC attack is differentiated from that of application-layer DDoS attacks and flash crowds and it is demonstrated why solutions for such problems are not viable for detecting a FRC attack. (2) A methodology is described that incorporates three detection metrics that together exploit the self-similarity [27] of site-level web usage. (3) Three attack scenarios are provided that test the proposed detection metrics

and provide extensive experimental results.

The rest of the chapter is organized as follows. Section 5.3 provides the background for the cloud and utility model. The FRC attack, threat model, cloud web server profiling, and related works are thoroughly discussed in Section 5.4. In Section 5.5, the datasets that is used for both training and detection are described. The detection metrics are described in Section 5.6 and the attack scenarios used to test the given detection mechanisms are described in Section 5.7. Experimental evaluation is provided in Section 5.8 and the FRC attack is discussed in context of a flash crowd in Section 5.9. Finally, Future Work and the Conclusion are presented in Sections 5.10 and 5.11 respectively.

5.3 Background

This section provides the background and context useful for the remainder of the chapter.

5.3.1 Cloud Computing and FRC Actors

As illustrated in Figure 5.1, the following roles are defined in the context of cloud computing to provide a consistent reference to the actors that play a role in a FRC attack.

- *Cloud Service Provider (CSP)* - The CSP (e.g. Amazon EC2, Microsoft Azure, or Rackspace) offers consumer-provisioned and metered computing resources that can be leased for flexible time durations.
- *Cloud Consumer* - The cloud consumer is a person or organization that employs the services of a CSP and is financially responsible for resource consumption. The cloud consumer also plays the dual role of the victim.
- *Client* - The client is a legitimate user that requests web content offered by the cloud consumer.
- *Attacker* - Although a FRC attack is ultimately carried out by one or more attack clients, the attacker (e.g. bot master) is the mastermind that orchestrates the FRC attack among the attack clients.

- *Attack Client* - The attack client is a malicious user (e.g. bot) that fraudulently consumes resources offered by the cloud consumer.

5.3.2 Cloud Utility Pricing Model

The utility pricing model is the offering of computing, bandwidth, and storage resources as a metered service. In some respects, the utility model is similar to traditional tariffs public utilities charge for resources like electricity, because both computing oriented and public utility oriented revenue is based on the quantity of resources consumed by their customers. Unlike the electricity networks however, the information services offered by a CSP are accessible through an open global network (i.e. the Internet).

Table 5.1: Amazon EC2 Data Transfer Pricing Metrics for US East(Virginia) as of January 2012

Data Transfered Out	
First 1 GB/Month	$0.00 per GB
Up to 10 TB/Month	$0.12 per GB
Next 40 TB/Month	$0.09 per GB
Next 100 GB/Month	$0.07 per GB
Over 150 GB/Month	$0.05 per GB

Cloud consumers only pay for the CSP resources they use and only for the time they use them. Although there are many metered services provided by CSPs, the focus of this paper is on the data transferred out of a CSP environment. To illustrate the actual costs of data usage in the cloud, Table 5.1 provides a summary of the costing metrics for Amazon's Elastic Compute Cloud (EC2) platform [4]. Under this pricing model and bounded by a Terms of Agreement, the cloud consumer is responsible for all data usage regardless of the intent of the requesting client.

Given that an average web page - including both primary and secondary objects - is 320KB in size [95], Figure 5.2 enumerates the accumulated daily cost of data usage when applied to Table 5.1 for a website that experiences 1, 5 and 10 requests per second respectively over a billing period (typically a month). Even on the order of 5 requests per second, data usage -

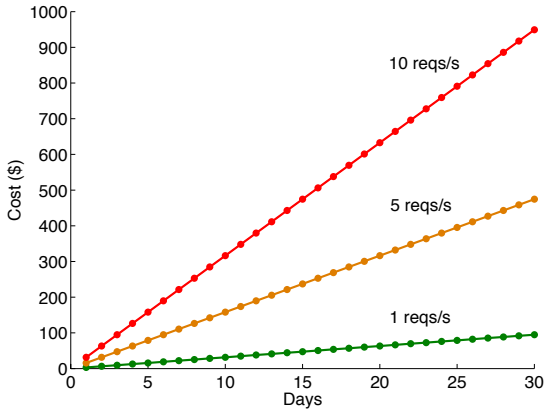


Figure 5.2: Accumulated Data Usage Costs

consumed maliciously or not - amounts to a significant monthly cost for the cloud consumer.

5.4 FRC Attack

This section provides the description for the FRC attack including a threat model, the profiling of cloud-based web servers, and related works in context of the described attack.

5.4.1 Threat Model

Similar to those that carry out DDoS attacks, a FRC attacker considered in this work is financially motivated by a fee for services rendered, by an extortion payment, or has an enjoyment-based intrinsic motivation [69]. Although in practicality an attacker could carry out a FRC attack from any Internet-connected device, for the purposes of this work it is assumed that the attacker has amassed a botnet of sufficient size and capability to easily perform a FRC attack.

In order to achieve the greatest cost burden, the attacker is interested in maintaing a sustained attack over a long duration of time. This is best accomplished by avoiding aggressive behavior and instead subtly inflating resource consumption in order to avoid detection. Ideally for the attacker, the size of the resource consumption push is large enough to achieve the cost burden objectives in the least amount of time. Attack optimization is not in scope of this work.

The target of the attacker is a public-facing web application or website hosted in a public CSP environment that is governed by a utility compute pricing model. Content from the victim's website is public-facing and intended to be viewed by the general public. While authentication such as passwords could be used to de-annoynimize the requesting clients and limit the amount of publicly exposed web content, such access control is considered counter-productive for distributing public-facing web content. Furthermore, the victim does not make use of reverse Turing tests [116] to differentiate between humans and attack clients. Turing tests have been found to be unreliable and have been circumvented by mechanical Turks [18] and puzzle breaking schemes [126, 131]. Moreover, requiring a casual web user to solve a graphical puzzle to simply view a webpage is counter to the goals of the target web application as some users are unable or simply unwilling to deal with the hassle [87, 97].

It is also assumed that the web server on which the victim's website resides is properly patched, adheres to a well-managed security policy and is buffered from the Internet by a firewall that employs security best-practice filtering rules. Furthermore, all requests generated by the attacker adhere to all protocols specifications. The only exploitable vulnerability considered in this threat model is the utility compute model.

5.4.2 Cloud Web Server Profiling

Measuring the capacity of web servers hosted in the cloud is a key component for better understanding the vulnerability that the utility pricing model presents to the cloud consumer. Enumerating cloud server capacity highlights the significant gap that exists between normal web server request rates and cloud web server performance capacity. To enumerate this gap, a number of experiments were performed on Amazon's EC2 - actual cloud infrastructure. The objective of these experiments was to determine the requests per second capacity of the various EC2 server offerings for a range of average web page sizes. The three standard server instant sizes offered by EC2 and thus considered in this work are presented below [5]. Included in this description are the hardware and software specifications of these instances. Although EC2 does offer other high-performance instances with increased memory or CPU capabilities, the experiments performed were considered to be general use-case scenarios. It is assumed and

expected that such high-performance instances would be able to handle an equal amount or more requests than the standard server instances presented.

Small Instance

1.7 GB memory

1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)

I/O Performance: Moderate

SUSE Linux Enterprise Server 11 Service Pack 1 basic install, EBS boot, 32-bit architecture with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.0, PHP 5.3, Ruby 1.8.7, and Rails 2.3.

Large Instance

7.5 GB memory

4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each)

I/O Performance: High SUSE Linux Enterprise Server 11 Service Pack 1 basic install, EBS boot, 64-bit architecture with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.0, PHP 5.3, Ruby 1.8.7, and Rails 2.3.

Extra Large Instance

15 GB memory

8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each)

I/O Performance: High SUSE Linux Enterprise Server 11 Service Pack 1 basic install, EBS boot, 64-bit architecture with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.0, PHP 5.3, Ruby 1.8.7, and Rails 2.3.

The web servers examined in these experiments were used to gauge the capacity of a web server (i.e. FRC target) hosted in the cloud. To provide consistency throughout the many experiments, the requesting host (i.e. FRC attacker) was deployed on a small EC2 instance. The purpose of the attack host was to simulate an aggregate user request base (legitimate or malicious) by enacting requests for a web document on the target web server by utilizing the Apache web server benchmarking tool [6]. To determine the capacity of requests per second for

a given instance, requests were generated from the attack host located in the same geographical region (i.e. East 1-d) as the target host in order to reduce the effects of network latency, among other factors, in this measure.

To accurately gauge the average size of the web documents being publicly hosted by the target server, which is necessary to provide a realistic experiment, Table 5.2 provides a selection of web metrics calculated by Google [95] describing websites and web documents as found on the World Wide Web (WWW).

Table 5.2: Google Web Metrics

Metric	Top Sites	All Sites	Description
Pages	380 million	4.2 billion	Number of pages analyzed
GETs			Average number of GETs per page
Mean	42.14	43.91	
Median	33	37	
Network Size (KB)			Average size per page
Mean	312.04	320.24	
Median	176.23	177.47	
KB per GET			Average size per GET
Mean	7.32	7.19	
Median	2.36	1.93	

Given that the average webpage, encompassing both primary and secondary requests (Section 4.4.1) is 320KB in size and that the average web page consists of nearly 44 web elements, then, as shown in Table 5.2, the average object size, for all websites on the WWW, is 7.19KB. Based on this average measure, web pages of both 1KB and 2KB smaller and larger than 7.19KB were constructed and hosted on the target web server to provide a range of measures for capacity testing. To enact requests from the attacking host, the Apache benchmark tool was utilized by requesting 10,000 requests in total, allowing for 20 concurrent requests for the attack host at any one time.

To give context to the experimental results and to provide a reference for the normal requests per second observed by an actual web server, consider Iowa State's public web domain. During a 12-week period in the Fall of 2011, the *iastate.edu* domain experienced, on average, 5.63 total

request per second (including both primary and secondary requests). Similarly, the NASA data set [82], used throughout this work, experienced a rate of 0.72 total requests per second over a two-month period. Although these websites are not indicative of all sites on the WWW, they provide a baseline with which to compare to the capacity of EC2 web server instances. To reiterate, the objective of these tests is to measure the capacity of a web server hosted in the cloud to illustrate the gap between the actual request intensity experienced by a common web server and the upper bound of what a cloud instance is capable of handling. The results are expected to show that cloud-based web servers are sufficiently over provisioned and thus create a significant gap with which an attacker can wage a FRC attack without being detected by current DDoS detection mechanisms.

Depicted in Figure 5.3 are the average results of five test iterations for the previously described experiments. For each a small, large, and extra large EC2 instance, requests per second measures were taken for web document sizes of 5.19, 6.19, 7.19, 8.19, and 9.19 KB. As it can be seen, for the average web document size found on the WWW (i.e. 7.19KB), even a small instance running as a web server is capable of servicing 2000 requests per second, three orders of magnitude more than the rate generated by the described reference websites (i.e., iastate.edu and NASA). For large and extra large instance, this capacity, in terms of requests per second, increases by over 200%.

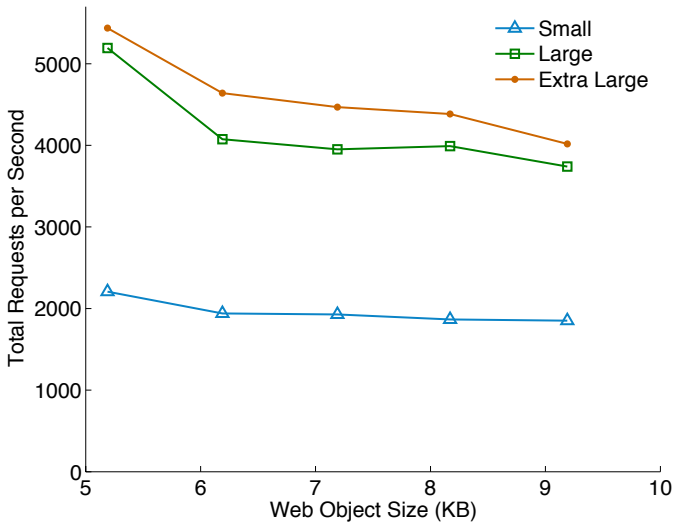


Figure 5.3: EC2 Capacity Profiling: All Requests

As the result of presenting many measures in this work in terms of primary requests, Figure 5.4 illustrates the translation from all requests in Figure 5.3 to the expect primary requests per the statistics shown in Table 5.2. In essence, this figure provides the measure of actual user generated requests, which is calculated be 1/44 of all requests.

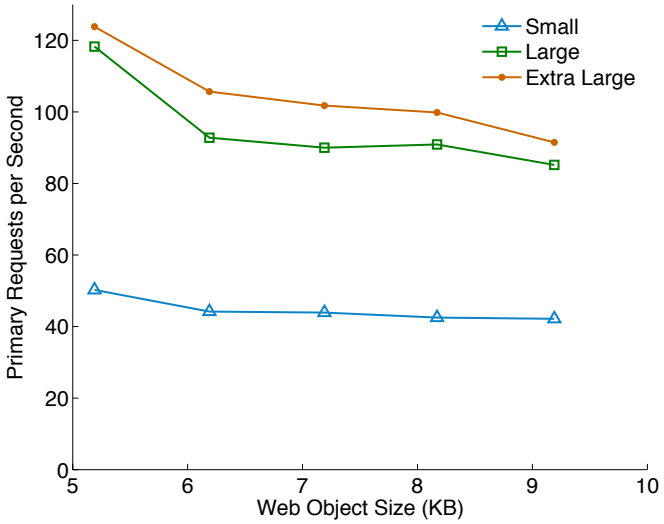


Figure 5.4: EC2 Capacity Profiling: Primary Requests

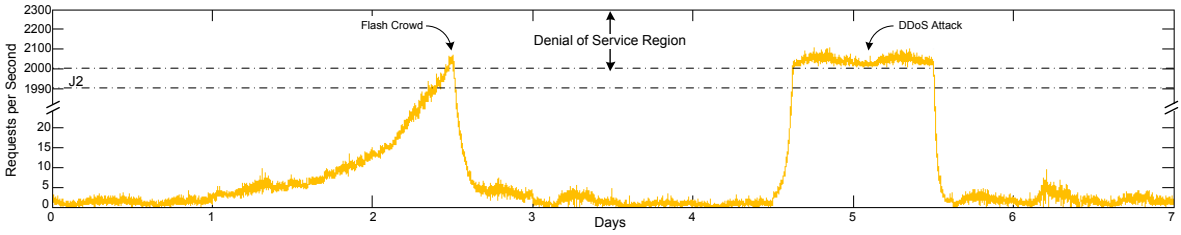
As demonstrated by these experiments, modern day web server capacity in the cloud is sufficiently capable of handling the request loads for an average sized web site. Even a one or two order of magnitude increase in each of the example request rates provided would still leave sufficient headroom for a small server instance hosted in the cloud to operate without diminishing the QoS for the end user. Given this gap between normal request rates and the capacity of cloud infrastructure, the FRC attack will be described in detail and an explanation will be provided as to why current DDoS detection and attribution methodologies do not apply to a much more subtle FRC attack.

5.4.3 FRC Attack Description

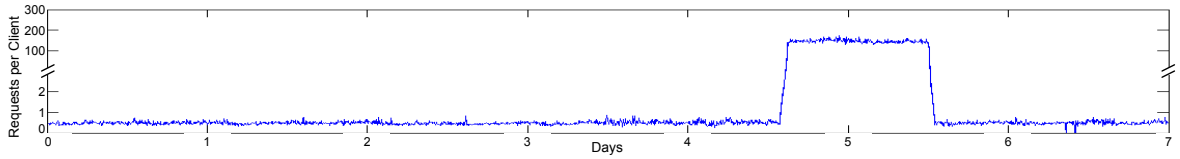
Together flash crowds and application-layer DDoS attacks provide an apt comparison with which to illustrate the request dynamics and subtleties of a FRC attack. Like flash crowds and application-layer DDoS attacks, an attack client involved in a FRC attack also victimizes the application layer by way of making protocol adherent requests. Differentiating a FRC attack is that the aggregate requests do not degrade availability of a given site. A FRC attack is intended to run-up the data usage cost for the victim by consuming bandwidth in excess of normal usage. In order to provide a clear description of the FRC attack and to explain why currently proposed application-layer DDoS solutions are largely ineffective for mitigating FRC attacks, Figure 5.5 illustrates the key differences between flash crowds and application-layer DDoS attacks.

5.4.3.1 Flash Crowds and DDoS Attacks

Depicted in Figure 5.5a is a graph of the all requests per second for a week-long web trace from a busy NASA web server [82]. Based on observations of flash crowds and application-layer DDoS attacks [56, 124], request dynamics for each of these two web phenomena were synthesized and interwoven into the web log trace in order to provide an illustration for the following example. For the purposes of continuity, each event was synthetically generated within the same web trace over non-overlapping periods. Figure 5.5b depicts the corresponding ratio of requests per client for these two events and interleaving request activity. Together Figures



(a) Requests per Second



(b) Requests per Client

Figure 5.5: Flash Crowd and Application-layer DDoS Comparison

5.5a and 5.5b provide the necessary context to describe and differentiate the FRC attack from flash crowds and application-layer DDoS.

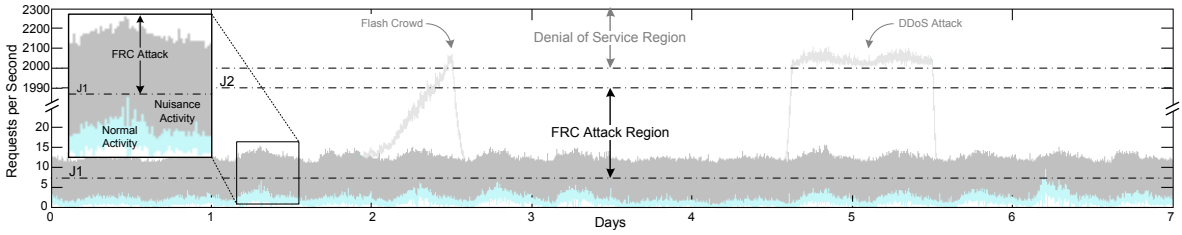
Beginning at Day 1 and culminating at Day 2.5, a flash crowd was generated to depict the request dynamics of this event. Flash crowds are defined as a significant number of legitimate clients simultaneously requesting web content from a given site. Often triggered by a major news [111] or sporting event [56], the individual *clients* that participate in this phenomenon are not malicious in nature. However, the aggregation of their actions is detrimental to the availability of the site under consideration. As shown in Figure 5.5b, the resulting per client request rate throughout the duration of the flash crowd remains consistent with normal usage despite the dramatic increase in requests per second. Clients that unintentionally participate in a flash crowd continue to behave in a manner that is consistent with the normal user behavior and thus during a flash crowd the request volume maintains the same proportion with the number of clients accessing the site. As observed in [56], request per client rates have actually been shown to decrease during a flash crowd as clients react to the diminished QoS of the system.

Although consequences may be similar, application-layer DDoS attacks provide a stark and malicious contrast to flash crowds. Attack clients participating in a application-layer DDoS attack utilize legitimate HTTP GET requests in excessive volumes to overwhelm a

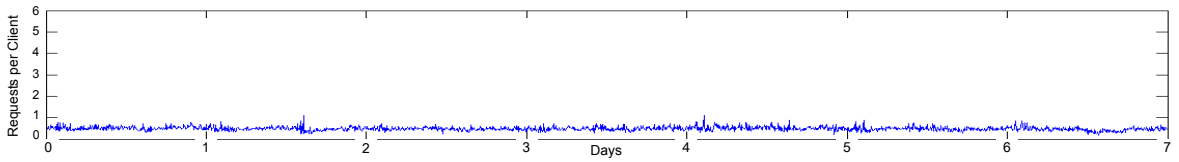
target's resources. Consequently, legitimate clients are subjected to a significantly diminished or unavailable service due to the reduction in throughput and increased server response latency. As shown in Figure 5.5a, beginning at approximately Day 4.7 and terminating on Day 5.5, a DDoS attack is an abrupt increase in the amount of requests per second experienced by a site - similar in the peak request rate to that of a flash crowd. However, as depicted in Figure 5.5b, the dramatic increase in the request per second rate during a DDoS attack is attributed to the increase in the per client request rate, which is significantly different from that of a flash crowd or normal usage. DDoS attacks are characterized by relatively few clients requesting web content at a very high rate or many malicious clients requesting at a lower, but still at a saturating rate. In either of these two cases or anywhere in between, the per client request rate deviates greatly from that of normal activity. Thus in an application-layer DDoS attack, the per client request rate increases in direct proportion to the increase in the per second request rate. It is this very observation that has been used as a key differentiator in proposed application-layer DDoS detection methods that seek to to distinguish flash crowds from application-layer DDoS attacks [56, 70, 97, 120].

Given the two web phenomena in Figure 5.5a and within this context, as the aggregate request intensity increases significantly (three order of magnitude more than is shown) above normal activity - whether it be malicious or not - this intensity significantly degrades the QoS of the web server. The scale chosen in Figure 5.5a was based on the experimental results from Section 5.4.2 for a small server instance on the EC2 platform and for an average file size of 7.19KB. As shown, such a server is a conservative estimate given modern computing capacity in the cloud. Aggregate request intensity depicted between 2000 and 2300 requests per second - labeled as the Denial of Service Region - is where the QoS of the system degrades to the point where the *clients'* user-experience is intolerable. Just below the Denial of Service Region is the threshold labeled as J2 on Figure 5.5a and is the point at which application-layer DDoS attacks and/or flash crowd mitigation solutions begin to sense and activate in order to quell the pending threat of unavailability [58, 120]. The threshold J2 is an approximate sensitivity benchmark indicating the state-of-the-art of reliable detection.

Considering request intensities above J2 is outside the scope of this work as it is assumed



(a) Requests per Second



(b) Requests per Client

Figure 5.6: FRC Attack Illustration

that current solutions would be able to mitigate such anomalous request activity. Instead the focus of this paper is on fraudulent request intensities below J2. Below this point, currently proposed mitigation solutions are not enabled because the QoS of the system is not significantly affected and would inevitably trigger on false positives which lead to the unnecessary rejection of legitimate *clients*.

5.4.3.2 FRC Attack

In comparison to the previously described flash crowd and application-layer DDoS attack, a FRC attack is a much more subtle event. For a FRC attacker to be successful, they need not deny service to the system but instead consume enough bandwidth to incur a fraudulent and unsustainable cost onto the cloud consumer. Building on Figure 5.5, Figure 5.6 depicts the request per second (Figure 5.6a) and requests per client (Figure 5.6b) characteristics of a FRC attack. As was the case in Figure 5.5, a FRC attack was synthesized and interwoven with that of the same week-long NASA web trace. In Figure 5.6a the request per second behaviors of both the flash crowd and DDoS attack were persevered to provide contrast and emphasis on the subtleness of a FRC attack.

The lowest layer of web traffic illustrated in Figure 5.6a is that of *normal activity* as generated by legitimate clients - labeled in the callout. Normal activity represents the requests

generated by the legitimate clients that frequent and interact with the site under consideration. The requests generated by legitimate *clients* provide value to the cloud consumer that is necessary to justify the costs of hosting the web application in a CSP environment.

In the presence of a FRC attack, data usage consumed by fraudulent requests is additive to that of normal activity. At the lowest intensities of illegitimate use, the resulting costs do not significantly impact the financial well-being of the cloud consumer. Such request intensities are considered *nuisance activity* and can be expected given the ever present noise originating from the Internet.

As malicious resource consumption intensifies, it will reach a point where the cost to the cloud consumer is no longer trivial. This threshold is denoted as J1 on Figure 5.6a. As shown in the callout, request volumes in excess of J1 are considered to be that of a FRC attack. The requests performed by attack clients in a FRC attack have no intention of providing value to the site and seek to only incur fraudulent charges. For a FRC attacker to be successful, they need only to increase the amount of data usage consumed over a given period of time in comparison to that of normal activity. Such an increase in request volume can be produced by malicious clients utilizing published web usage distributions [7, 75]. Thus for attack clients to increase the data usage for a given site by over 500% - shown as request activity above the J1 threshold in Figure 5.6a - the malicious request dynamics need not exhibit a higher than normal per-client request ratio as shown in Figure 5.6b nor exhibit highly aggressive request patterns on which many application-layer attack detection and attribution mechanisms depend to distinguish malicious clients from that of normal clients.

As a recap, the FRC attack region is defined as the aggregate request intensities between thresholds J1 and J2 on Figure 5.6a. At the lower end of the continuum (J1), fraudulent requests begin to financially impact the cloud consumer. As FRC attack intensities increase, the resulting cost becomes more significant. At point J2, a FRC attack exhibits behavior characteristics that are comparable to a DDoS attack and thus detectable. It is assumed that for request intensities above J2 that current DDoS mitigation strategies would be able be effective. As FRC attack detection improves, the J2 threshold will drop and ideally reaching J1 in the future. It is significant to note the substantial gap between J1 and J2 (i.e. the FRC

attack range) on Figure 5.6a. As shown, an FRC attacker can achieve success by consuming 10 reqs/s to nearly 1980 req/s more than normal traffic while avoiding extreme usage patterns that would lead to detection. While the significant over provisioning of web servers is helpful in many contexts, when a utility pricing model is applied, the gap between normal activity and J2 presents and unaddressed vulnerability of the current cloud computing infrastructure.

5.4.4 Related Work

The catalyst for this work is directly attributed to security professional Christopher Hoff who first described the Economic Denial of Sustainability (EDoS) attack [51] on the cloud model as a manipulation of the utility pricing model that results in unmanageable costs for the cloud consumer. The premise of the EDoS attack, as described by Hoff, is for an attacker to exploit the horizontal scaling capability [77] of both compute and storage resources hosted in the cloud. This present work instead explores a much more subtle threat model in which the FRC attacker does not seek to exploit resource costs in an overwhelming manner, which would require a significant amount of malicious resource consumption over a short period of time. Instead FRC is a slow-and-low attack over a longer duration of time that exploits resources much more gradually. Additionally, when taken a face value, an EDoS attack is a class of attacks of which FRC, DDoS, and click fraud are members for they inevitably deprive a victim of economic value through direct costs, loss of business, or damaged reputation. In the context of cloud computing, the FRC attack as described in this chapter and in this work, aptly defines the subtle exploitation of resources governed by a utility pricing model.

The related bodies of research that have bearing on the FRC attack can be broadly categorized as detection schemes that seek to differentiate flash crowds from application-layer DDoS attacks and attribution methodologies that aim to distinguish legitimate clients from application-layer DDoS attack clients. Each of these bodies of work and subsequent papers will be examined in the context of the FRC attack.

5.4.4.1 Detection

To better understand the nature of abrupt changes in HTTP GET request rates, Wen et al. [120] proposed an entropy-based solution that compares the ratio of observed source IPs and target webpages. The results reported in [120] are consistent with the analysis in Section 5.4.3.1. While their observations may lead to the classification of dramatic increases of web traffic, as evidenced in Section 5.4.3.2, FRC attack clients need not exhibit the same aggressive nature of attack clients that participate in a DDoS attack to accomplish their goal. As a result, entropy of per document request diversity is unlikely to differentiate a FRC attack from any non-aggressive increase in normal activity. Similarly, solutions that trigger on the slope of request intensity [70] are also likely to be ineffective at detecting FRC attacks based on the same principles.

In contrast to the request intensities of the user population, other solutions have sought to differentiate flash crowds from application-layer DDoS attacks based on the source IP distribution of the requesting clients [56, 70]. The hypothesis is that malicious clients participating in a DDoS attack will originate from IP clusters not previously experienced by the site. However, as stated in [58, 97, 124], such a solution is not reliable as it is likely that geographically distributed attack clients will originate from similar IP prefixes and clusters belonging to legitimate clients.

5.4.4.2 Attribution

To exploit the limitations of automated attack clients, Kandula et al. [58] proposed a scheme that makes use of graphical puzzles to differentiate human initiated requests from those automatically generated. Although effective, such an approach is not generally regarded as a practical solution [87, 124] to prevent an application-layer DDoS attack. Graphical puzzles are out scope in the FRC attack context, see the Section 5.4.1. Similarly, the use of honeytokens [40] - human-invisible objects imbedded in web content - has also been proposed to identify automated clients [87]. The premise of this methodology is that only attack clients that haphazardly traverse web content will request invisible objects and thus be flagged as malicious. Despite positive results, such a solution is only applicable to a very limited threat

model.

In addition to visual approaches, others have sought to differentiate attack clients by scrutinizing the transitional probabilities between successive client requests [87, 124]. Albeit a promising solution for a subset of attack scenarios considered for the FRC attack, this approach can be defeated with little effort by an attacker.

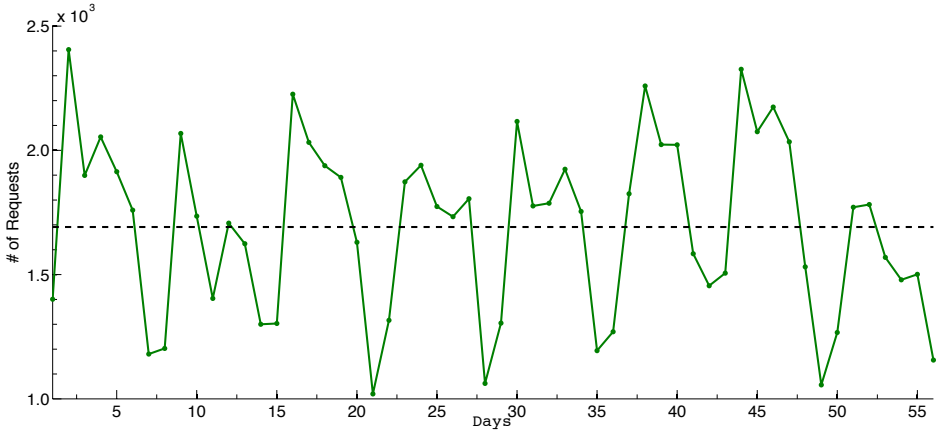
In the context of the FRC attack, the greatest downfall of many of the previously proposed application-layer DDoS mitigation solutions is that they do not activate until the victim web server is under extreme duress [58, 120]. Even if activated, many solutions are predicated upon the fact that attack clients exhibit anomalous inter-session and inter-request arrivals times in comparison to that of request dynamics from normal activity [56, 97, 120]. While effective for detection of an attack client participating in a DDoS attack, as was discussed in Section 5.4.3.2, such solutions are not viable for the detection of a FRC attack. This is due the lack of necessity of the attack client to request higher than average request intensities.

In fairness to the mentioned authors, their research focus was not on detecting a FRC attack. Although each of the works cited are successful at accomplishing their own respective objectives, these solutions are not directly applicable to the detection or attribution of a FRC attack.

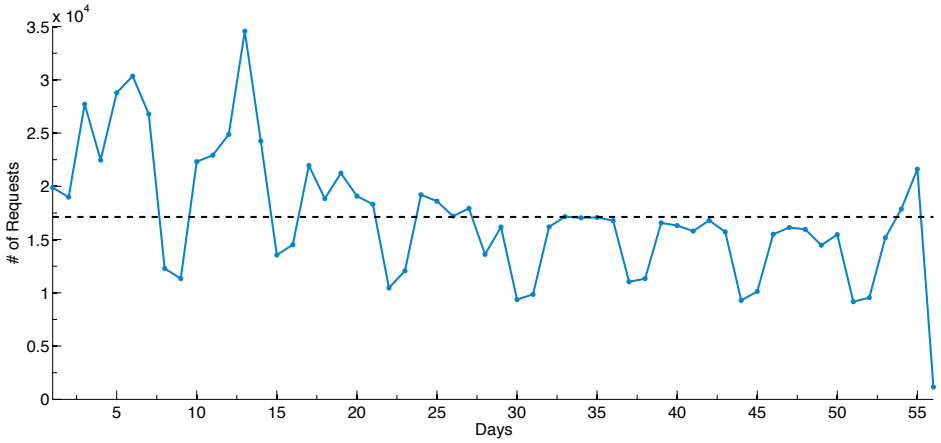
5.5 Dataset Description

Two 56-day web log traces are used for both training and testing of the proposed detection metrics presented in this chapter. The first dataset is a web trace from ISU's Department of Electrical and Computer Engineering's public-facing web server (referred to as ECpE). The second dataset is a historical web trace from a busy NASA web server (referred to as NASA) that has been made available to researchers [82]. The request per day plots for each dataset in Figure 5.7 illustrates the fluctuation in the day-to-day request volumes experienced by each respective site. The daily volume variation is potentially as much as 100% and that characteristic has driven the search for detection metrics with limited sensitivity towards request volumes.

Although both datasets follow a similar cyclical pattern, the nature of the datasets - one being a departmental website and the other being a public website for a U.S. Governmental



(a) ECpE Requests



(b) NASA Requests

Figure 5.7: Detection Dataset Description

agency - provide sufficient contrast in the request volumes (Table 5.3) and user demographics to test the generality of the proposed detection scheme.

For the purposes of this paper, each of the datasets have been reduced from their original form. As is typical in the retrieval of web content, primary client-issued requests invoke secondary (or inline) requests to retrieve supporting web content such as embedded pictures, scripts and style sheets. The focus of this chapter is on the primary requests invoked by the user and, as a result, all secondary requests were removed from the datasets.

Table 5.3: Description of Experimental Datasets

Metric	ECpE	NASA
Days	56	56
Total Requests	94 681	977 930
Max Reqs/Day	2 405	34 566
Min Reqs/Day	1 020	9 171
Clients	14 891	130 128
Sessions	39 929	304 207
Avg Daily Req Variation	56.87%	74.91%

5.5.1 Attack Scenario

To quantify a FRC attack, consider the NASA dataset. At an average request volume of 17 463 requests per day, for a FRC attacker to increase the request volume by 100% with a 1000-client botnet - modest by current standards - each attack client would only be required to request 18 web documents per day. At such a small and low intensity quota, individual clients will have no problem blending in with the request dynamics of normal clients. It follows that to increase the web traffic by 1000% or 10 times the average cost, attack clients would need to request 180 web documents per day. Even at a request volume 10 or 100 times greater than that of normal activity and given the capacity of cloud computing infrastructure, the target site would be far from experiencing request volumes that would result in a DDoS. The hypothesis this chapter seeks to verify is that despite the malicious request volumes, the structure of the accumulated malicious requests will deviate sufficiently from normal activity to be detectable.

5.6 Detection Metrics

There are two objectives for detection mechanisms that have been identified in this section. The first is to take advantage of the consistency and self-similar nature of aggregate web activity. The second is to be able detect the presence of resource usage fraud within an inspected dataset. In other words, the detection techniques characterize the level of consistency of request semantics relative to normal activity as opposed to factoring in volatile request volumes. This section describes three detection metrics that are used together to detect FRC attacks.

5.6.1 Zipf's Law

The properties of Zipf's law [132] provide a useful metric for measuring relative frequency and self-similarity of web document popularity. Although Zipf's law has been used as an anomaly detection metric to detect accounting fraud [52] and blog spam [81], the predominant application of Zipf's law in relation to web requests has been in the modeling and formulation of web caching schemes [65]. Based on these related works, this section provides a description of Zipf's law and description of the means by which Zipf's law is applied in order for it to be a key metric for the detection of FRC attacks.

Given a web server log that is composed of client-initiated request records for a website hosting multiple web pages, let N be the total number of distinct web pages requested for a given period of time. Let f_i define the frequency - the number of times the page was requested - for each $i \in N$ web pages. Based on the request frequency of each respective page, let the pages be ranked as a list in descending order based on their popularly such that the most frequently requested page assumes the rank of one and the i^{th} page is the i^{th} most requested page. Based on this ranked list of aggregate client requests, for Zipf's law to apply, the request frequency for the i^{th} most popular web page is inversely proportional to the rank of the page and is represented as follows:

$$f_i \propto \frac{1}{i} \tag{5.1}$$

As shown in past research efforts [14, 17, 125], aggregate web requests do not strictly follow Zipf's law, but instead more generally conform to a Zipf-like distribution in which the frequency for the i^{th} most popular page is a power-law function such that [125]:

$$f_i \propto \frac{1}{i^{\psi}} \tag{5.2}$$

For Zipf's law to directly apply to a distribution of web requests, ψ would assume the value of unity. Instead, when a web request distribution is plotted on a log-log scale as a rank-frequency plot, the resulting slope (ψ) of the best-fit regression line for the Zipf-like

distribution is typically negative and approximately unity. The Zipf-like distribution for the ECpE and NASA datasets are shown in Figures 5.8 and 5.9 respectively.

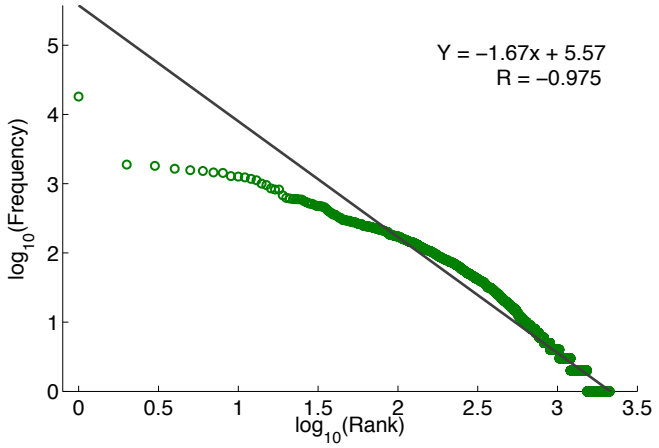


Figure 5.8: Zipf-like Distribution for ECpE Dataset

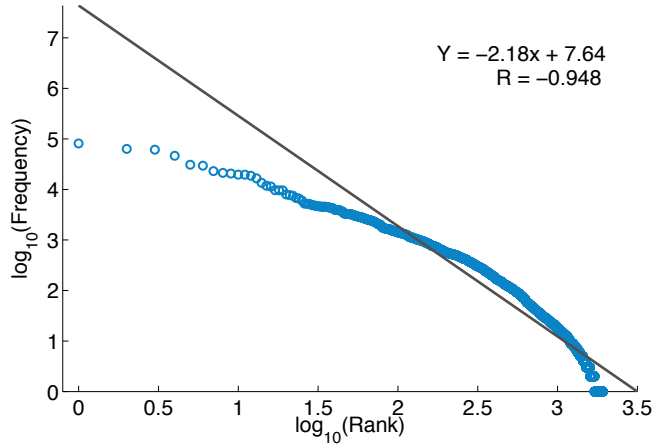


Figure 5.9: Zipf-like Distribution for NASA Dataset

In the context of a FRC attack, the primary interest is the set of web documents that represent the majority of overall data usage. As shown in Figure 5.10, a three-day trace from the NASA dataset, 10% (154 web pages) account for 90% of all web requests. Furthermore, it is observed that to the left of this point the plotted data conforms to that of a power-law distribution, and to the right of this point the plot diverges and is considered to be an exponential tail. As a result, the detection metric presented in this paper focuses only on the

top_k documents that summarize 90% of all web requests for the respective dataset.

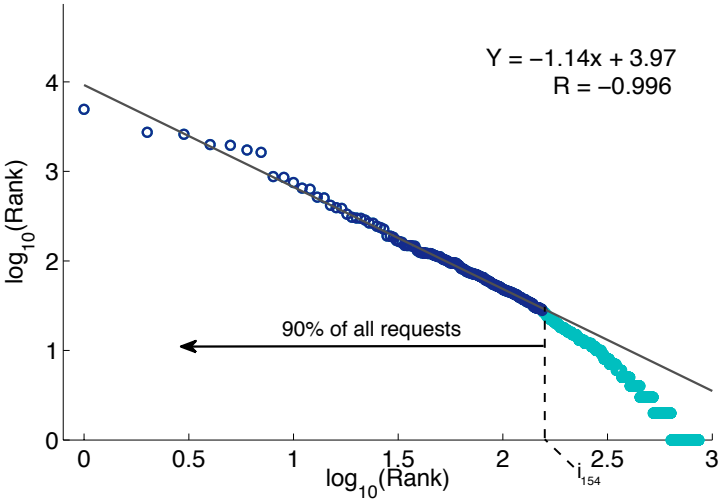


Figure 5.10: Calculation of Zipf-like Regression Slope

As shown through experimentation (Section 5.8.1), the regression slope of 90% of web requests has provided a consistent measure of aggregate use patterns that conform to the principles of self-similarity. The hypothesis of this detection measure is that it would be a tall order for an attacker to initiate requests that conform with the Zipf-like distribution of the overall normal activity of a particular website. This detection metric is utilized by computing the 90% regression slope of a sample data set and determining if the slope falls within a tolerance interval relative to normal activity. To succeed in foiling this detection technique, the attacker would need to be privy to the website's usage patterns. This measure will be referenced as the *Zipf value*.

5.6.2 Spearman's Footrule

The Spearman's Footrule distance [32] is a non-parametric measure of similarity, or lack thereof, between two ranked lists. Although this measure has been used in other web-related contexts such as measuring the accuracy of predicting individual session n-grams generated from Markov models [10], Spearman's footrule appears not to have been used as a metric for the detection of anomalous web usage. While the more general use of the Spearman's Footrule

is to find an aggregated ranking that minimizes or maximizes the distance between two ranked lists, for the purposes of this paper, the proximity between two ranked lists will instead be considered as it is a more appropriate measure for the objectives of anomaly detection. Thus the greater the Spearman proximity between two top_k ranked lists, the more similar the two lists are in respect to each other.

Building on the notation established in [10], the Spearman's Footrule proximity is defined as follows: Given two ranked top-k lists L_1 and L_2 as inputs, with each list containing k entries, let L_1 be the reference list that is assumed to be the ground truth and L_2 be the comparison list that is highly likely to be a *partial list* in comparison to that of the reference list. The term *partial list* refers to a comparator list being of equal cardinality as the reference list, but its membership is missing one or more elements present in the reference list. To obtain the ranking of a list item $i \in L_1$, we define the function $f(i)$ and similarly $g(i)$ for $i \in L_2$. In function $g(i)$ if $i \notin L_2$, then the subsequent ranking is assigned that of a location parameter $l = k + 1$ [105], where $k = |L_1|$. Given these preliminaries, the Spearman's Footrule proximity is defined as follows:

$$S(L_1, L_2) = 1 - \frac{\sum_{i \in L_1} |f(i) - g(i)|}{k(k+1)} \tag{5.3}$$

In the case that both ranked lists are identical, the Spearman proximity would result in the value of one and in the case of two disjoint lists the Spearman proximity value would be zero. In order to provide a measure of similarity or proximity instead of a measure of difference or distance, the normalized summation in Equation 5.3 is subtracted from 1. This measure of similarity between a training and test dataset is known as the *Spearman value*.

5.6.3 Overlap

To complement the Spearman value, the Overlap between the reference list L_1 and the comparator list L_2 is measured to provide a broad indication of the similarity between the training data and that of test data respectively and is referred to as the *Overlap value*. The lists L_1 and L_2 are defined in the same manner as in Spearman's Footrule. As experienced

through experimentation, when used separately the Spearman and Overlap values are prone to misclassifications but when used together the synergy between the two detection metrics provide a consistent non-parametric measure of relative document popularity. The Overlap value is defined as the percentage of items in the comparator list L_2 that appear in the reference list L_1 such that:

$$O(L_1, L_2) = \frac{|L_1 \cap L_2|}{|L_1|} \tag{5.4}$$

Thus given two equal sized lists, the Overlap between them is the cardinality between the intersection of L_1 and list L_2 divided by the number of elements in the reference list.

5.6.4 Detection Training and Testing

As a product of the detection metric calculations from the training data, a statistical tolerance interval [86] for each metric was constructed to serve as a detection criterion. The use of a tolerance interval in this context bounds a confidence interval that covers a percentage of an assumed normal sample population with which one would expect an individual sample value to fall such that $\bar{X} \pm g_{(1-\alpha,p,n)}s$ where $1-\alpha$ is the confidence, p is the percent coverage of the sample, n is the sample size and s is the standard deviation.

$$g = \sqrt{\frac{(N-1)(1+\frac{1}{N})z^2_{(1-p)/2}}{\chi^2_{\gamma,N-1}}} \tag{5.5}$$

For each of the three previously described detection metrics, a 80% tolerance interval summarizing 99% of the training data were calculated. When a detection measure is outside the tolerance interval, the understanding is that a FRC attack has been detected. Likewise, if the detection measure falls within the tolerance interval the understanding is the test sample is likely clear of an attack. In order to measure quality of the detection metrics, the false positive rates (FPRs) and false negative rates (FNRs) were computed. A false positive (FP) is when the detection measure falls outside the tolerance interval but the data is free of a FRC attack. Furthermore, a false negative (FN) is when the detection measure is within the tolerance interval, but the data contains an attack.

Given many different attack strategies, the Spearman, Zipf and Overlap detection metrics are individually prone to unacceptably high FPRs and FNRs. However, when used in combination, the complementary nature of the metrics yield a reliable approach for the detection of FRC attacks.

5.7 Attack Description

The quality of a detection scheme is a product of the environment in which it is tested. This section describes three probable attack strategies that are used for the experimental evaluation of the described detection metrics and are likely to be carried out by a FRC attacker. While attack patterns can be crafted in a way that will be detected with a high measure of success, the attack strategies chosen for the purposes of this chapter were done so to highlight both the strengths and weaknesses of the proposed detection metrics. In order to challenge the methodology, the attacker is given a great deal of latitude with which to mount a FRC attack.

5.7.1 Attack Assumptions

Overall, it is assumed that the attacker has full knowledge of the web logs before the onset of a FRC attack. The three proposed attack strategies depict different ways with which the attacker could use such information to fraudulently consume web content. A simplifying assumption has been made in that the training data used to prime the detection metrics is free of attack behaviors.

5.7.2 Attack Strategies

5.7.2.1 Random Attack

In the random attack strategy, the attacker creates an unsorted list of the distinct web pages requested by the legitimate clients as seen in the web logs. To mount an attack, the attacker randomly chooses web documents from this list. To introduce a degree of variability into this attack pattern, the *percent* parameter - denoted Pct - is used to narrow the scope of the content requested by the attacker. For example, for a Pct=0.5 the attacker would randomly

generate fraudulent requests from a list of 50% of the known web content previously requested from the target site. Although the random attack is the least sophisticated of the proposed attack patterns, it perhaps is the easiest attack to employ and can be performed without prior access to the web log.

5.7.2.2 Heavy-Hitter Attack

The heavy-hitter attack simulates a scenario in which the attacker seeks to consume the most frequently requested documents experienced by the target site. To carry out the heavy-hitter attack, the attacker creates a ranked list from requested documents from the web log. From the ranked list of the most popular web pages, similar to that of the random attack pattern, the attacker randomly requests content from the top Pct of available web documents. For instance, for a ranked list of 100 web documents and Pct=0.1, the attacker would only request the 10 most popular web pages from the victim's site. For a Pct=1.0, the heavy-hitter attack strategy would be no different from that of the random attack strategy. More cunning than the random attack, the heavy-hitter attack seeks to blend into the normal activity of a site by requesting the most frequently requested documents.

5.7.2.3 Trace-Driven Attack

As shown in Chapter 4, the most cunning attack strategy is a trace-driven attack in which the attacker replays request sequences extracted from that of the observed training data. Similar to the two previous attack scenarios, the Pct parameter is used to limit the scope of the attacker to a smaller percentage of known sequences of requests. Given a web log that has 500 distinct request sequences, with a Pct=0.40, the attacker would only have access to 200 of the request sequences. The trace-driven attack scenario is very similar to that of a prescribed-path attack strategy - an attack in which the attacker guesses popular web request sequences with which to mount an attack - however, such an attack is more formidable as the attacker is not required to guess, but only replay known request patterns.

5.8 Experimental Evaluation

To explore the robustness of the proposed detection metrics, this section provides the experimental results performed to calculate both the FPRs and the FNRs for each of the datasets and the attack strategies presented in Section 5.7.2.

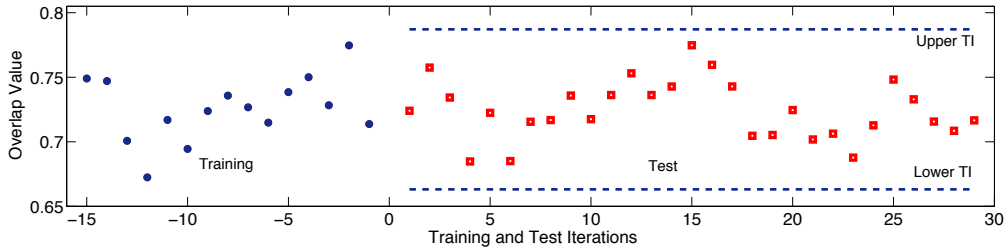
5.8.1 False Positive Rate Results

To calculate the FPRs for the proposed detection scheme, each of the two 56-day datasets were partitioned into separate training and testing datasets. Using a window of three days, 15 detection metric measurements were taken and tolerance intervals summarizing 99% of the data with a 80% confidence - chosen to minimize the false-positives and false-negatives - were calculated for each of the three detection metrics. With the remaining dataset partitioned for testing, the respective detection metrics were calculated and applied to each of the three tolerance intervals.

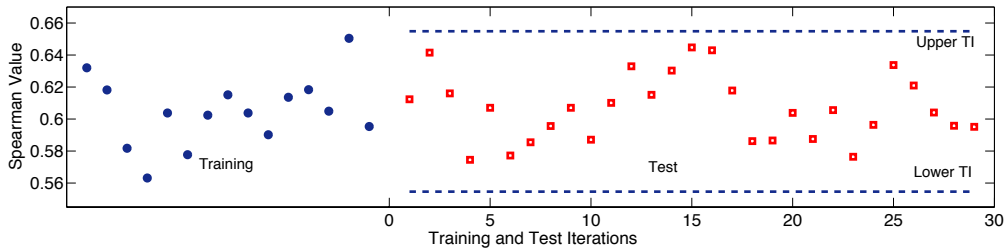
For a test iteration to be deemed a true negative, the test window under consideration, known to be free of an attack, is required to satisfy each of the three proposed metrics (i.e., Zipf, Spearman, and Overlap) by producing a measure within the respective tolerance intervals. A single breach of a tolerance interval for any of the three tests would be considered anomalous and thus register as a false positive. As shown in Figures 5.11b, 5.11a, and 5.11c for the ECpE dataset, the detection schemed failed to yield a single false positive for any of the test iterations and thus had a FPR = 0.0% for the stated parameters.

Similarly for the NASA dataset, as seen in Figure 5.12, the detection scheme performed equally well resulting in zero false detections and likewise had a FPR=0.0%.

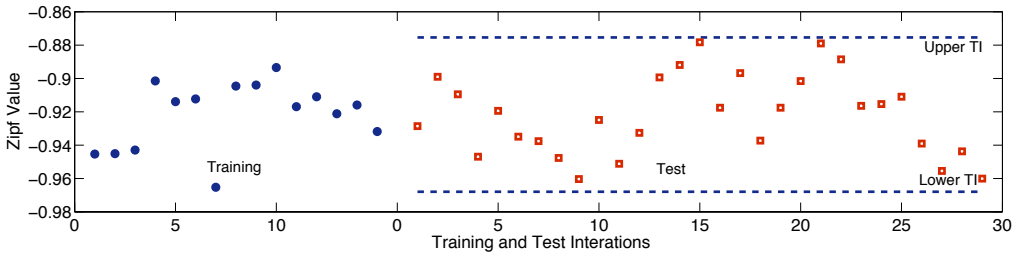
As shown in both Figures 5.11 and 5.12, the proposed detection metrics exhibit a consistent and reliable measure of normal activity for each of the respective datasets. These results supports the hypothesis that the self-similarity of the aggregate web traffic for a given website can be exploited despite the variance in the day-to-day request volume in the training and test datasets (Table 5.3).



(a) Overlap



(b) Spearman's Footrule

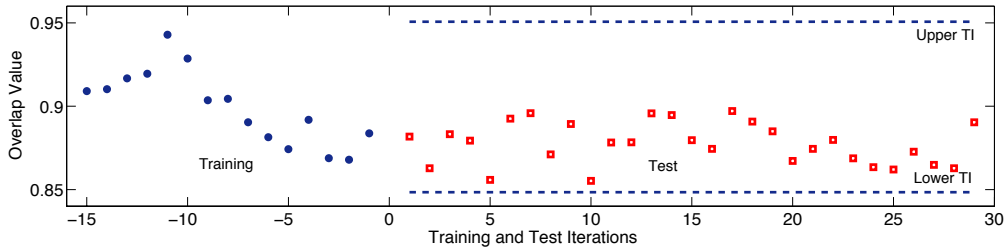


(c) Zipf's Law

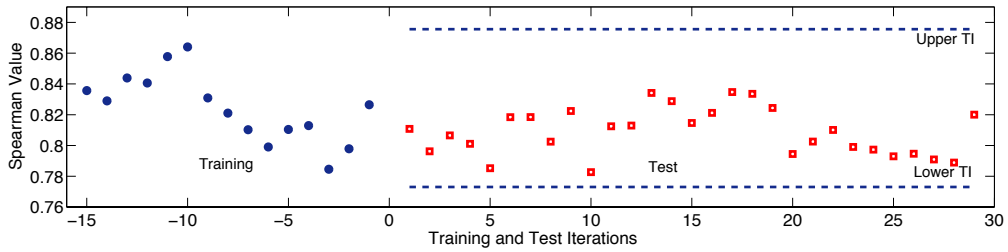
Figure 5.11: ECpE False Positive Confidence Intervals

5.8.2 False Negative Rate Results

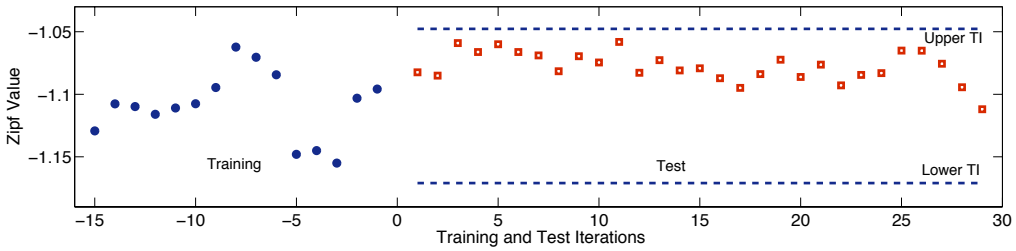
Given the three attack strategies, the objective of this section is to determine the percentage of data usage above that of normal activity an attacker can consume without being flagged as malicious. For each attack strategy and dataset (ECpE and NASA), a table presenting the results for each of the test cases is provided. For each of these tables, the rows denote the percentage of data usage consumed in excess of normal activity - labeled as *Att*. For example, *Att*=0.4 would represent a test case in which the attacker consumed 40% more data than that of normal traffic over the given training window size. The column headings represent the percent parameter - labeled *Pct* - that was described in the context of each of the attack strategies discussed in Section 5.7.2.



(a) Overlap



(b) Spearman's Footrule



(c) Zipf's Law

Figure 5.12: NASA False Positive Confidence Intervals

Different than the FPR results presented in Figures 5.11 and 5.12, in the context of detecting a FRC attack, for an attacker to succeed and not be detected (i.e. produce a FN), the result of an attack must produce measures that are within the trained tolerance intervals for all three detection metrics. If one of the three metrics falls outside of a tolerance interval, just as was the case in the calculation of the FPRs, the test iteration would be considered an attack and thus a TP.

5.8.2.1 Random Attack Strategy

In the random attack strategy, the attacker randomly requests a percentage (Pct=0.1-1.0) of the known requested web pages and uses this knowledge to mount a FRC attack. Presented

in Table 5.4 are the FNRs for the ECpE dataset. Based on these results, little success is achieved by the attacker. At the worst, the attacker is able to consume 30% of data usage in addition to that of normal activity with a success rate of 17%. In the context of the ECpE dataset 30% more than normal would amount to a subtle 0.006 requests per second in addition to normal traffic - a non-aggressive rate as far as of DDoS detection schemes are concerned. For the remaining tests cases, the attacker was only able to achieve minor success consuming 10% more bandwidth than normal, which also translates to nuisance activity as found on Figure 5.6a.

Table 5.4: ECpE: Random Attack Strategy - False Negative Rates (%)

Att/Pct	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.07	0.00^{2}	0.02	0.01	0.02	0.03	0.02	0.08	0.09	0.09
0.2	0.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.3	0.17^{1}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.5	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Examining the first highlighted cell in Table 5.4 (Pct=0.1, Att=0.3) - denoted by the superscript 1 - shows a minor ineffectiveness of the detection scheme against the random attack strategy. As shown in Figure 5.13, individually each of the three detection metrics produces a high number of FNs. However, when used together as was previously described, one can see the effectiveness of the overall detection scheme presented in this chapter. For this particular attack scenario, the random attack pattern was able to stay within all three of the calculated tolerance intervals for 5 out of 29 test iterations producing a FNR of 17%.

To contrast the previous results, the second highlighted cell in Table 5.4 (Pct=0.2, Att=0.1) - denoted by the superscript 2 - depicts a case in detection methodology that failed to record a single FN, which overall was the predominant trend for the random attack strategy. In this

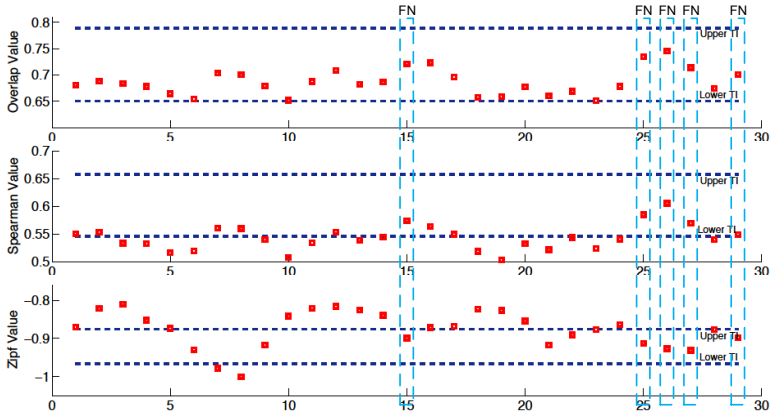


Figure 5.13: ECpE: Random Attack Scenario - Pct=0.1, Att=0.3

case, as can be seen in Figure 5.14, the random attacker pattern consistently altered the Zipf value to a point where each test iteration produced a TN measure and thus an overall FNR = 0.00%.

The general trend across all test iterations was that the Overlap and Spearman values produced more FNs measures as the diversity of requests used in the attack increased. Randomly requesting webpages from an equally likely distribution caused the Overlap and Spearman metrics to remain consistent in terms of accuracy and completeness with that of normal traffic but inflated the volume of each of the requested documents. As a result, the predominant metric that caused the random attack pattern to fail (i.e. produce TNs) can be attributed to the Zipf metric. Especially as seen in Figure 5.14, the biasing of request volume caused by the random request pattern eroded the expected proportionality between consecutively ranked documents, which resulted in a decrease in the slope of the regression line for the resultant Zipf-like distributions.

The NASA dataset (Table 5.5) also achieved similar results for the random attack strategy. At best, the attacker was only be able to consume 10% more than normal data usage with a success rate of 0.03%, which would easily fall into the classification of nuisance activity as well.

Similar to the ECpE dataset, a test iteration of Pct=0.1 and Att=1.0, highlighted in Figure 5.15, outputs very similar results for the NASA dataset. Due to randomness and low diversity of requests enacted by the attacker, both the Overlap and Spearman values are skewed enough

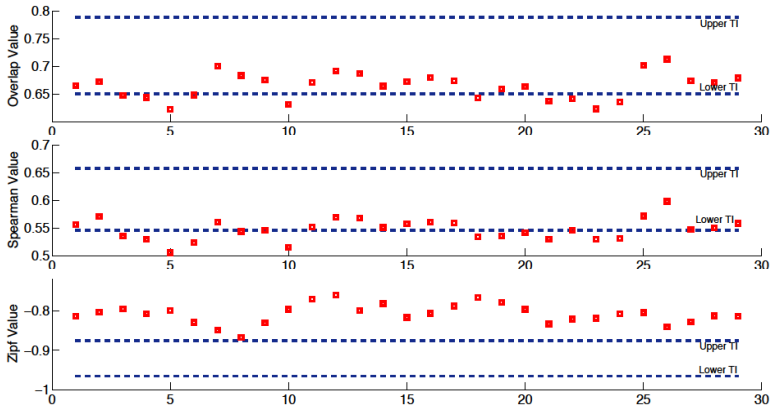


Figure 5.14: ECpE: Random Attack Scenario - Pct=0.2, Att=0.1

Table 5.5: NASA: Random Attack Strategy - False Negative Rates (%)

Att/Pct	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.00	0.00	0.00	0.01	0.01	0.03	0.03	0.03	0.03	0.03
0.2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.0	0.00[1]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

to point where they registered intermittent TP and FN marks. Consistent with the ECpE dataset, the random request attack pattern is unable to preserve the relative volume of requests between consecutively rank documents and thus the Zipf metric produces a large number of TP measures. As depicted in Figure 5.15, the Zipf metric recorded a single FN given 29 test iterations. To the strength of the detection scheme, both the Overlap and Spearman metrics registered as TP for the single Zipf metric that registered as a FN and thus overall produced a TP classification.

The results of the random attack strategy demonstrates that although this attack strategy would be the simplest of the three strategies for an attacker to perform, the attacker would

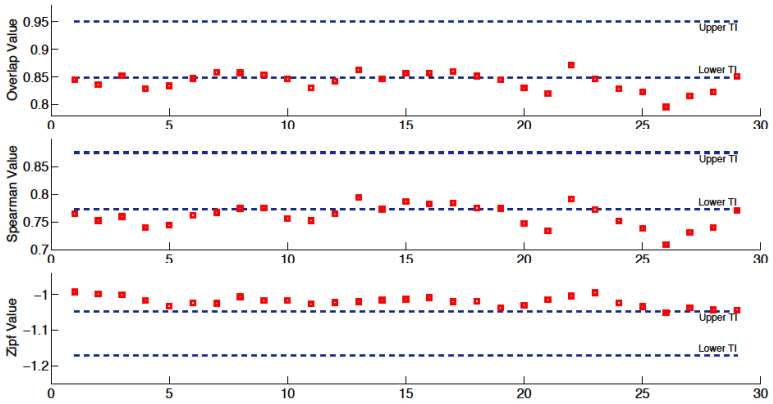


Figure 5.15: NASA: Random Attack Scenario - Pct=0.1, Att=1.0

not be able to consume enough data to mount a substantial FRC attack. For the attacker to achieve success against the presented detection methodology, it is clear the attacker must attempt to preserve the Zipf-like distribution produced by normal users.

5.8.2.2 Heavy-Hitter Attack Strategy

Similar to the random attack strategy, for the heavy-hitter strategy the attacker also randomly draws page requests from a list, but in this case the list is sorted based on the popularity of the given web document. This attack strategy tests the detection scheme's ability to withstand an attack that attempts to mask its occurrence by drawing on the most popular documents as requested by legitimate clients. In this context, Pct=0.1-1.0 in Tables 5.6 and 5.7 represents the pool of most popularly requested documents the attack clients draw from. For instance Att=0.1 would be the top 10% of documents and 100% would be all documents and thus no different than that of a random attack strategy.

As seen in Table 5.6 (Pct=0.3) and Table 5.7 (Pct=0.1), the heavy-hitter strategy exploits a weakness of the detection methodology. Based on the natural composition of the two datasets and the resulting tolerance intervals, the stated Pct values represent a range of the top_k documents that can be fraudulently consumed, with varying success, while still satisfying each of the three detection metrics.

In Table 5.6 for Pct=0.3, Att=0.1-1.0, it can be seen that the detection scheme breaks down

Table 5.6: ECpE: Heavy-Hitter Attack Strategy - False Negative Rates (%)

Att/Pct	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.28	0.55	0.45	0.29	0.07	0.01	0.02	0.07	0.14	0.15
0.2	0.00	0.29	0.41	0.20	0.03	0.00	0.00	0.00	0.00	0.00
0.3	0.00	0.07	0.38	0.08	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.00	0.01	0.33	0.03	0.00	0.00	0.00	0.00	0.00	0.00
0.5	0.00	0.00	0.28	0.06	0.00	0.00	0.00	0.00	0.00	0.00
0.6	0.00	0.00^1	0.30^2	0.05	0.00	0.00	0.00	0.00	0.00	0.00
0.7	0.00	0.00	0.29	0.06	0.00	0.00	0.00	0.00	0.00	0.00
0.8	0.00	0.00	0.30	0.06	0.00	0.00	0.00	0.00	0.00	0.00
0.9	0.00	0.00	0.28	0.03	0.00	0.00	0.00	0.00	0.00	0.00
1.0	0.00	0.00	0.26	0.03	0.00	0.00	0.00	0.00	0.00	0.00

for a specific range of generated requests. Figure 5.16 displays the detection results for the first highlighted cell in Table 5.6 (Pct=0.2, Att=0.6) - denoted by the superscript 1 and in the column or attack percentage adjacent to the trouble area. From these results, it is quite clear, just as it was for the random attack strategy, that the Zipf metric prohibits the attacker from conducting a FRC attack unnoticed. Because the heavy-hitter attack strategy only initiates requests from the most frequently sought documents, the effect of the attack on the Spearman and Overlap metrics is beneficial to the attacker. The heavy-hitter attack pattern does not significantly change the rank of the most requested documents, thus the large number of FNs. Requesting the most popular documents does, however, alter the Zipf value to the point where it is considered anomalous.

Examining the individual detection metric values for the highlighted cell (Pct=0.3, Att=0.6) adjacent previously discussed cell - denoted by the superscript 2 - presents a clearer picture of why the detection scheme failed for the range of requested documents where Pct=0.3. As shown in Figure 5.17, the attack pattern was able to produce a number of measures within the predetermined Zipf tolerance interval. Coupled with the failure of both the Overlap and Spearman metrics for this particular range of requested documents, the attacker was able to achieve some success defeating the detection scheme. Comparing the trend in Zipf values between Figures 5.16 and 5.17 shows that the for a Pct=0.3 in Figure 5.17 the Zipf value

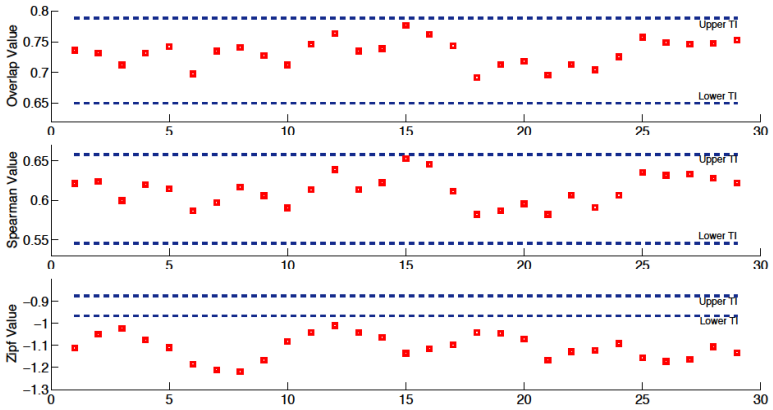


Figure 5.16: ECpE: Heavy-Hitter Attack Scenario - Pct=0.2, Att=0.6

produces measures that are in the range of the Zipf tolerance interval while in Figure 5.16 the Zipf values are clearly larger than the stated tolerance interval. Although not presented, the Zipf metric values for the test case where Pct=0.4, Att=0.6 exhibits a consistent increase in the Zipf value, moving swing of calculated measures sufficient below the tolerance interval to produce very little FNs. Thus given the heavy-hitter attack scenario, there is a natural deficiency that, if found, could be exploited by an FRC attacker. However, despite this short coming, by only being able to achieve a success rate of 25% (i.e. a FNR of 25%), the amount of requests generated by such an attack, from an accumulated cost perspective, would be insignificant and is thus labeled as nuisance activity.

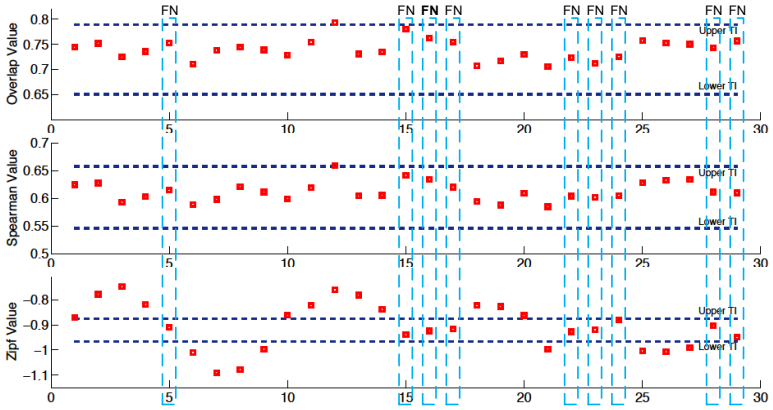


Figure 5.17: ECpE: Heavy-Hitter Attack Scenario - Pct=0.3, Att=0.6

For the NASA dataset, the overall results from the heavy-hitter attack scenario differ slightly in Table 5.7 due to the contrast in composition of the two datasets but present the same general trend as previously discussed for the ECpE dataset. The range of documents that is able to defeat the Zipf metric for the NASA dataset occurs when Pct=0.1.

Table 5.7: NASA: Heavy-Hitter Attack Strategy - False Negative Rates (%)

Att/Pct	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.90[1]	0.00[2]	0.00	0.00	0.05	0.09	0.05	0.06	0.06	0.05
0.2	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.3	0.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.79	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.5	0.57	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.6	0.55	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.7	0.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.8	0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.9	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.0	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 5.18 presents the results for the first highlighted cell (Pct=0.1, Att=0.1) in Table 5.7 - denoted by the superscript 1. Similar to measures produced in Figure 5.16, the necessary structure in the data logs has been preserved to keep the Zipf value within the calculated tolerance interval. At an Att=0.1, the heavy-hitter attack scenarios appears to have bested the detection methodology. However, at Att=0.1, even with a success rate of 90%, the attacker is only consuming 10% more bandwidth than normal and thus the impact over the overall attack does not enter into the FRC attack range. For Pct=0.1, the observed trend in Table 5.7 is that as the attack intensity increases the effectiveness of the attack result decreases.

Consistent with the results for the ECpE dataset, an increase in the percentage documents requested causes the heavy-hitter attack strategy to fail. The test iterations for the adjacent cell (Pct=0.2, Att=0.1) - denoted by the superscript 2 - in Table 5.7, shown in Figure 5.19, presents a consistent result of the decrease in the Zipf value as Pct increases.

In comparison to the random attack strategy, an attacker employing the heavy-hitter strategy is able to defeat the FRC detection scheme with a higher degree of success. Despite this

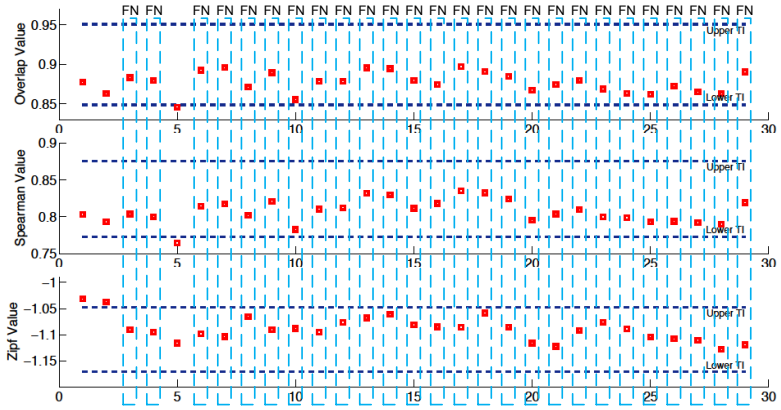


Figure 5.18: NASA: Heavy-Hitter Attack Scenario - Pct=0.1, Att=0.1

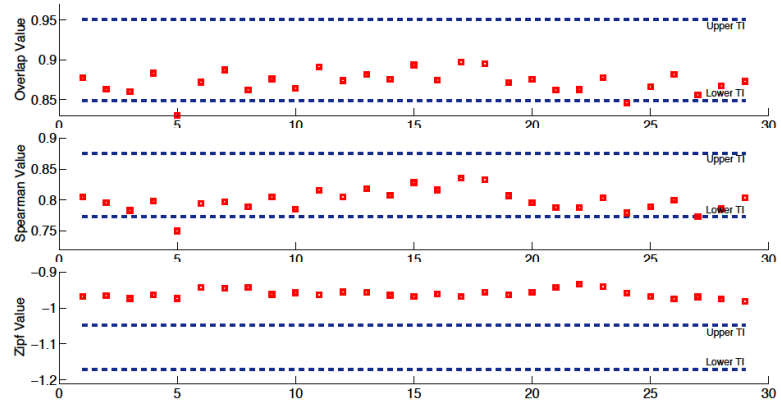


Figure 5.19: NASA: Heavy-Hitter Attack Scenario - Pct=0.1, Att=0.1

limited success however, the detection metrics are able to detect a vast majority of the attack test cases and not a single registered test case yielded a FNR that would result in a successful FRC attack.

5.8.2.3 Trace-Driven Attack Strategy

The third and most formidable attack scenario is the Trace-Driven attack strategy in which the attacker simply replays the request sequences of users that have already visited the site. Even without the assumption of allowing the attacker to have knowledge of the training logs, a trace-driven attack scenario can be easily carried out by an attacker by crafting potential

request sequences that mimic the web traversal patterns of a legitimate user interacting with the victim's site. Choosing highly likely request sequences, not requesting random and unpopular pages that defy the link structure of a given website, thwarts detection efforts that seek to identify malicious clients based on known transitional probabilities between successive requests [87, 124].

The objective of this attack strategy is to explore the diversity of request sequences needed and volume of web traffic that can be consumed without being detected as malicious. In this scenario, both the attack volume and percentage parameter differ from that of the two previous attack strategies. A trace-driven attack that makes between 10% to 100% more requests than that of normal traffic is too subtle to be detected and thus a range of 100% to 1000% is instead considered. Furthermore, enabling an attacker to replicate 50% to 100% of known sequences likewise produces unacceptably high FNRs. Therefore, the attacker's knowledge has been restricted to that of 5% to 50% of known sequences. For each of these two omitted ranges, these results are to be expected because the resulting consumption patterns within these ranges are no different from that produced by the site's normal clientele and daily fluctuations in request volumes.

To highlight the point of the FRC ineffectiveness against the trace-driven attack for the given ranges, consider the results in Figure 5.20 for an attack that consumes 1000% more requests than normal by replaying all available sequences observed (Pct=1.0, Att=10.0). Thus in many respects this test scenario generates a flash crowd, an event the detection scheme should not regularly detect. Even at 10 times the request volume of normal traffic, the detection methodology fails to detect this action consistently as a FRC attack, which, given the context, is the desired result.

The experimental results shown in Tables 5.8 and 5.9 depict the FNRs for synthesized attacks that ranged in excess between 100% and 1000% (Att=1.0-10.0) of normal activity and replayed 5% to 50% (Pct=0.05-0.50) of the known web request sequences. As can be seen by the trend of FNRs in Table 5.8 for the ECpE dataset, the more diversity and subtleness in request volume the attacker exhibits, the more difficult a FRC attack is to detect and thus the danger of a FRC attack. For an attacker to accomplish their goal, substantial financial damage

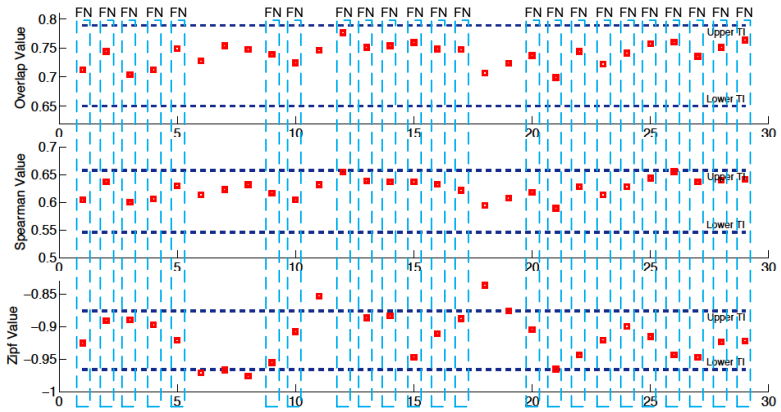


Figure 5.20: ECPE: Trace-Driven Attack Scenario - Pct=1.0, Att=10.0

can be inflicted by requesting less intense volumes over longer durations of time.

Table 5.8: ECpE: Trace-Driven Attack Strategy - False Negative Rates (%)

Att/Pct	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
1.0	0.00	0.07	0.22	0.37	0.48	0.53	0.56	0.56	0.57	0.66
2.0	0.00	0.00	0.03	0.13	0.25	0.37	0.40	0.47	0.57	0.55
3.0	0.00	0.00	0.00	0.08	0.18	0.30	0.39	0.44	0.56	0.54
4.0	0.00	0.00	0.00	0.07	0.17	0.24	0.36	0.44	0.44	0.51
5.0	0.00	0.00	0.01	0.07	0.14	0.17	0.31	0.39	0.46	0.53
6.0	0.00	0.00	0.00	0.05	0.11	0.22	0.36	0.37	0.46	0.49
7.0	0.00	0.00	0.00	0.06	0.09	0.20	0.30	0.37	0.45	0.48
8.0	0.00	0.00	0.00	0.05	0.07	0.16	0.26	0.33	0.45	0.48
9.0	0.00	0.00	0.00	0.03	0.07	0.13	0.25	0.32	0.47	0.48
10.0	0.00	0.00	0.00	0.05	0.06	0.17	0.31	0.36	0.40	0.47

Of the 39 939 observed request sequences for the ECpE dataset, 20% (i.e. column 2 in Table 5.8) represents 7 986 of those sequences. Likewise for the NASA dataset that consists of 304 207 distinct request sequences, 20% of these request patterns equate to 60 841 sequences. To put the less than desirable FNR results in Tables 5.8 and 5.9 into perspective, these results are predicated upon the fact that an attacker either has full access to the web logs or is able to accurately guess 8 000 or 61 000 web request sequences for each of the respective websites. Thus for less knowledgable attacker and given the proposed detection metrics and experimental results shown, it would be difficult for an attacker to mount a successful FRC attack and avoid

detection.

Table 5.9: NASA: Trace-Driven Attack Strategy - False Negative Rates (%)

Att/Pct	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
1.0	0.00	0.14	0.49	0.71	0.72	0.84	0.89	0.90	0.90	0.94
2.0	0.00	0.00	0.18	0.43	0.64	0.69	0.82	0.86	0.90	0.92
3.0	0.00	0.00	0.07	0.30	0.52	0.66	0.80	0.83	0.89	0.91
4.0	0.00	0.00	0.05	0.24	0.52	0.59	0.69	0.86	0.91	0.92
5.0	0.00	0.00	0.03	0.16	0.47	0.62	0.72	0.77	0.85	0.93
6.0	0.00	0.00	0.02	0.17	0.43	0.56	0.69	0.75	0.82	0.92
7.0	0.00	0.00	0.03	0.14	0.41	0.55	0.70	0.72	0.85	0.90
8.0	0.00	0.00	0.02	0.15	0.41	0.61	0.67	0.78	0.82	0.90
9.0	0.00	0.00	0.03	0.13	0.39	0.54	0.68	0.70	0.85	0.90
10.0	0.00	0.00	0.03	0.14	0.39	0.57	0.69	0.76	0.83	0.89

Based on these experimental results from the three presented attack scenarios, it can be seen that an attacker with knowledge of the web logs can perform a FRC attack with varying degrees of success given the detection metrics presented in this paper. Without the knowledge of the training data, the experimental results indicate that it would be very difficult for an attacker to mimic the aggregate web usage patterns to consume an appreciable amount of web data without being detected as malicious.

5.8.3 Self-Similarity and Consistency of Training Data

Like fractals, aggregate web traffic has been shown to possess self-similar characteristics [27, 29]. This scale-invariant quality of web data, more specifically the collective patterns of client HTTP requests, is relevant in context of FRC detection as it enables the presented detection methodology to be applied to various scales of time (i.e. training and test window sizes) based on the needs or requirements of a particular cloud environment. Although the experimental results shown in Sections 5.8.1 and 5.8.2 were performed for a training and test window size of three days, this parameter of the detection methodology is not fixed. Figures 5.21, 5.22, and 5.23 presents the experimental results for the NASA dataset for the testing portion of the detection methodology. By varying the training window size from one to seven days it can be seen that these results, the average Spearman, Overlap and Zipf values, exhibit statistical self-

similar qualities that lend this methodology to be applicable for more than a single training window size. While the average Spearman, Overlap, and Zipf values remain consistent, the tightening of the upper and lower tolerance intervals, for each of the presented figures, is in larger part due the decrease in variance of the metric measures as the training window increases.

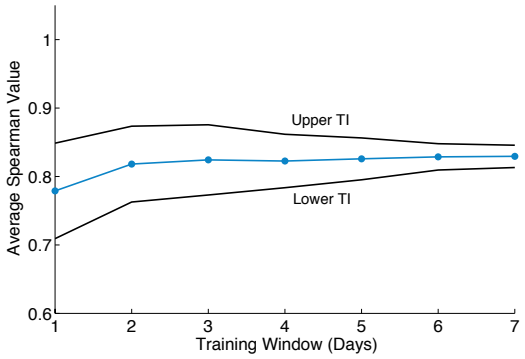


Figure 5.21: Spearman's Proximity: Self-similarity of NASA Dataset

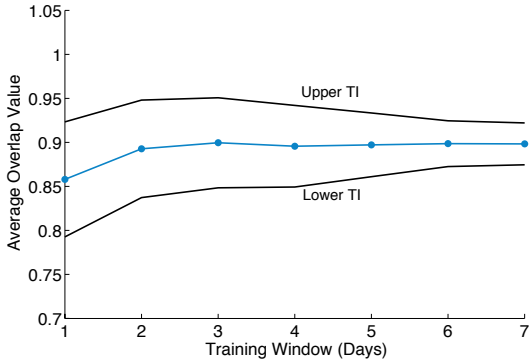


Figure 5.22: Overlap Value: Self-similarity of NASA Dataset

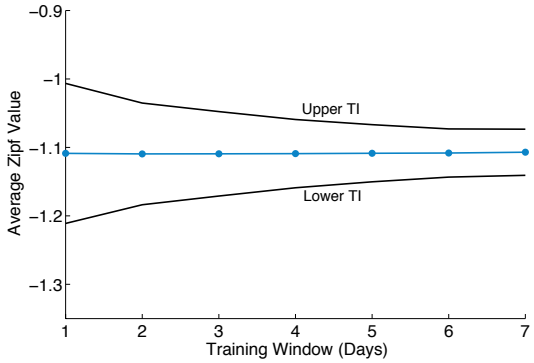


Figure 5.23: Zipf Value: Self-similarity of NASA Dataset

To supplement the previously described self-similar characteristics of the aggregate web requests, Tables 5.10 and 5.11 provide the FPR results for training window sizes of one to seven days to enumerate the consistency of the measures. The motivation for presenting the results of the three-day training window size in Section 5.8.1 was due to the fact that each of the training windows sizes for both datasets yielded a FPR of 0.00% for a tolerance interval with 80% confidence. While a 0.00% FPR is ideal, it is certainly not always obtainable given the inherent characteristics of the datasets and the inevitable give-and-take between FPRs

and FNRs. Furthermore, given this variance, a tolerance interval of 80% is considered in this context to be an aggressive measure for the purposes of anomaly detection, especially when the emphasis of the methodology is to reduce false positives in lieu of false negatives. As a result, lower confidence tolerance intervals are not expected to yield FPRs = 0.00% for all training window sizes. To show the contrast in experimental results for a more conservative tolerance interval, measures in Tables 5.10 and 5.11 were performed with a confidence of 95%.

Table 5.10: ECpE: Consistency of Detection Metrics Across Training Window Sizes

	ECpE Dataset FPRs			
Window Size (Days)	**Zipf**	**Spear**	**Overlap**	**Total**
1	1/37 = 2.70%	0/37 = 0.00%	0/37 = 0.00%	1/37 = 2.70%
2	1/33 = 3.03%	0/33 = 0.00%	0/33 = 0.00%	1/33 = 3.03%
3	0/29 = 0.00%	0/29 = 0.00%	0/29 = 0.00%	0/29 = 0.00%
4	1/25 = 4.00%	0/25 = 0.00%	0/25 = 0.00%	1/25 = 4.00%
5	0/21 = 0.00%	0/21 = 0.00%	0/21 = 0.00%	0/21 = 0.00%
6	1/17 = 5.88%	0/17 = 0.00%	0/17 = 0.00%	1/17 = 5.88%
7	0/13 = 0.00%	0/13 = 0.00%	0/13 = 0.00%	0/13 = 0.00%

To present a higher fidelity view of the data presented in Table 5.10 for the training window size of four days, Figure 5.24 depicts each of the three detection metrics, their corresponding tolerance intervals, and a visual indication of whether or not the test iteration registered as a false positive. Despite registering a FPR = 4.00%, which may be acceptable in some circumstances, it can be seen that the four-day training window size yields consistent measures for both the Spearman or Overlap values. The overall FPR can be attributed to the single FP measure recorded for the Zipf metric as seen in Figure 5.24. Referring back to Table 5.10, there is a general trend that the consistency of the Spearman and Overlap value and inconsistency of the Zipf value for the ECpE dataset is generally not unique to the training window size. Although the Zipf metric is more prone to erroneous measures, as there is more variance within

its measures, it is an absolutely necessary measure for the detection of the FRC attacks. While the Spearman and Overlap metrics can easily be defeated by an attacker requesting a handful of popular web pages (i.e. heavy-hitter attack scenario), the Zipf metric provides the necessary measure to ensure that ranked aggregate requests conform to the relative proportionality in volume as found in Zipf-like distributions.

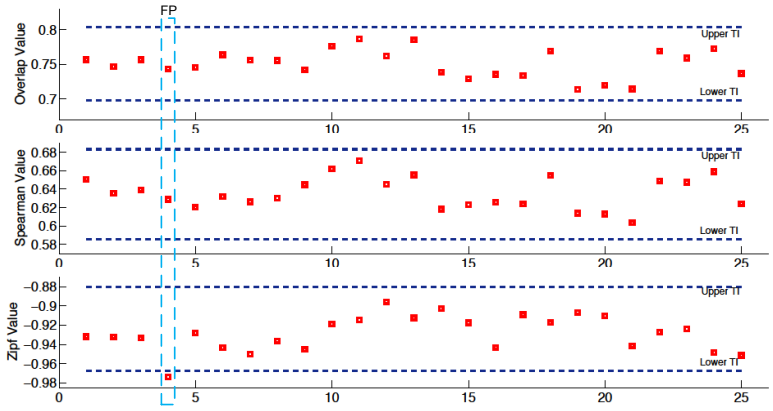


Figure 5.24: ECpE: FPR Results for Four-Day Windows Size

In comparison to the ECpE dataset, the measure of FPRs experienced by the NASA dataset for each of the detection metrics and across varying training window sizes, as seen in Table 5.11, yields more consistent and favorable marks. While specifying the root cause of these differences is difficult as there are many dissimilarities between the datasets that could effect this outcome, future research and access to a diversity of web logs will better help investigate the qualities of a dataset in respect to the consistencies (or inconsistencies) of its FRC detection measures. What is worth noting, however, is the applicability and accuracy of the given measures and the resultant FPRs for two very different websites and user bases.

As show in Table 5.11, a training window size of seven days for the NASA dataset also results in a FPR = 0.00%. As shown in Figure 5.25, although the window size has increased to seven days, due to the self-similar nature of the detection metrics, which can be observed for all graphed tolerance intervals, the resulting values provide a consistent measure with which to base an anomaly-based detection methodology.

While the FPRs will vary from website to website, the results reported show a promising

Table 5.11: NASA: Consistency of Detection Metrics Across Training Window Sizes

	NASA Dataset FPRs			
Window Size (Days)	**Zipf**	**Spear**	**Overlap**	**Total**
1	0/37 = 0.00%	0/37 = 0.00%	0/37 = 0.00%	0/37 = 0.00%
2	0/33 = 0.00%	0/33 = 0.00%	0/33 = 0.00%	0/33 = 0.00%
3	0/29 = 0.00%	0/29 = 0.00%	0/29 = 0.00%	0/29 = 0.00%
4	0/25 = 0.00%	0/25 = 0.00%	0/25 = 0.00%	0/25 = 0.00%
5	0/21 = 0.00%	0/21 = 0.00%	0/21 = 0.00%	0/21 = 0.00%
6	0/17 = 0.00%	0/17 = 0.00%	0/17 = 0.00%	0/17 = 0.00%
7	0/13 = 0.00%	0/13 = 0.00%	0/13 = 0.00%	0/13 = 0.00%

solution that is more general than the reported findings. The experimental results presented in this section demonstrate the applicability of the formulated detection metrics given a range of variability in the parameters that govern the experimental design of performing FRC detection. Furthermore, the characterization of the given datasets shows that the self-similar nature of web requests enables flexibility in the resolution of the training and testing window sizes.

5.8.4 Discussion

The detection of a FRC attack is fundamentally different than that of application-layer DDoS attack or systems compromise. With respect to a FRC attack, detection is not a binary measure in which the system is available or not, nor is it a question of whether a system's confidentiality or integrity have been compromised. Although ultimately desirable, the complete detection of a FRC attack is not necessary, just a sufficient amount that enables the entire FRC mitigation solution to relegate fraudulent use to that of nuisance activity. Reliable FRC attack detection thus answers whether a cloud consumer has been subjected to fraudulent resource use within a billing period or smaller unit of time. Building on this knowledge and as will be discussed in Chapter 6, once a FRC attack has been detected, it is then necessary to apply a

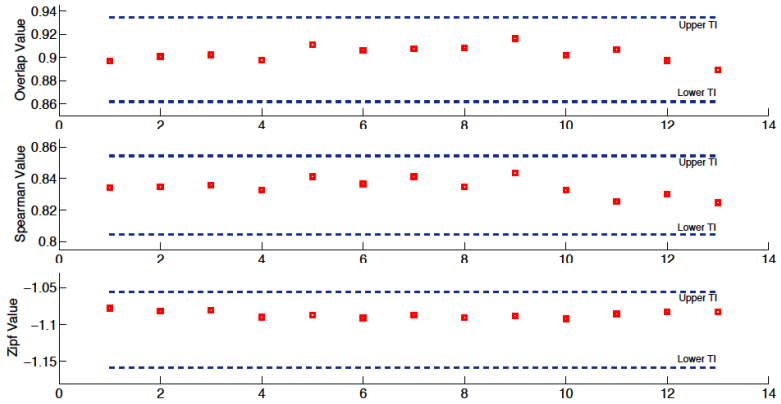


Figure 5.25: NASA: FPR Results for Four-Day Windows Size

FRC attribution solution to identify the malicious actors.

The methodology presented in this chapter relies on reviewing data over relatively large periods of time. The detection metrics were chosen for their ability to characterize aggregate data not for their computational performance. Although ideally FRC attack detection would be performed in real-time in a compute efficient manner, these qualities have lower priority relative to the need for low FPRs and FNRs. The FRC attack spans many days, weeks and possibly months, therefore available time and compute power are relatively abundant.

5.9 Flash Crowds and FRC Attacks

Relevant to the topic of FRC detection is to determine whether or not the presented FRC detection methodology detects flash crowds and, if so, whether or not this is a desirable quality. In this context, a flash crowd is a noticeable increase in the aggregate client request volume over a finite period of time. Flash crowds are typically short-lived anomalies triggered by an event that prompts a large number of unique clients to request a few web pages from a specific website. The abrupt and collective actions of a flash crowd can be troublesome to a web server as this behavior can inflict significant strain on available resources. Similar in nature to flash crowds are DDoS attacks that are composed of a relatively smaller number of malicious bot computers that each invoke a large number of requests. As has been previously discussed in Section 5.4.3.1, the telling characteristic that distinguishes flash crowds from DDoS attacks is

that the average per-client request rate for a flash crowd does not increase during such an event while the average per-client request rate increases significantly during a DDoS attack.

A FRC attack is comparable to that of a flash crowd in that the average per client request rate does not increase during an attack and thus such a measure cannot be used to make a distinction. However, there are other telling characteristics that can be used to differentiate these two events. As is central to the objectives of a FRC attack, the average per client request volume will increase over a given time period, which is different than that of a flash crowd. Similarly, the duration of a flash crowd is short-lived while a FRC attack as described in this work is to last weeks or months. Even if a FRC attack masks itself as a flash crowd, the overall impact of a short-lived event will not incur significant charges onto the cloud consumer.

The objective of the FRC detection methodology is to determine if increases in accumulated client traffic are consistent with that of what has been observed in the past or whether the request volume can be attributed to fraudulent consumption. To this affect, the FRC detection methodology should detect FRC attacks, DDoS attacks, and flash crowds as anomalies because each of these respective events are not indicative of normal traffic patterns. However, just because the presented FRC detection methodology deems each of these events as anomalous, this characterization does not infer that each of these events are a FRC attack. Once an increase in traffic have been determined to be anomalous, based on the duration of the event, constructs of the volume increase, and prior research, determining if an event is indeed a flash crowds, DDoS attacks, and FRC attacks is a matter of further analysis. The focus of this work is not on the events of extreme usage (i.e., flash crowds or DDoS attack) but instead on the task of differentiating very subtle increases in traffic from that of normal, which occurs on the other end of the attack spectrum. It is assumed that any resource usage in which the malicious clients behavior or request intensities deviate from normal will be trivial to detect and identify given the presented methodologies in this work. The challenging problem is detecting subtle attack clients attempting to mimic normal behaviors.

There is a historical precedence that when proposing a DDoS detection scheme that it should be able to accurately differentiate between DDoS attack and flash crowds, and, rightfully so, as the objectives of the website owner differ greatly depending on the nature of the spike in traffic.

In the case of a DDoS attack, the goal is to block or deny access to as many offending clients as possible to enable legitimate clients to access the given website. Handling flash crowds, on the other hand, it is the website owners objective to handle as many client requests as possible. The same motivation is true when differentiating between a FRC attack and an increase in traffic volume. In respect to the former, if the increased cost of traffic volume is not countered by added business value, then the long-term sustainability of operating in the cloud may not be feasible. As a result, a FRC attribution scheme would be used to identity and ultimately mitigation offending bots.

The operational purpose of the FRC detection and attribution methodologies should also be taken into account in this discussion. Like Intrusion Detection Systems (IDSs), the presented FRC detection methodology is a passive system that monitors traffic patterns for possible fraudulent or anomalous incidents and itself is not does not take action to stop a potential attack [103]. Similar to an Intrusion Prevention System (IPS), the FRC attribution methodology is tasked with identifying and blocking suspected clients based on the clients individual behaviors. In a sequence of events, a report of anomalous behavior from the FRC detection methodology would lead to the activation of the FRC attribution methodology. The reason the attribution methodology is not persistently engaged is that it would unnecessarily introduce false positive when the presence of an attack has not been shown.

Even if a FRC detection methodology errantly detects a flash crowd and engages the attribution methodology, such an action will have a minimum impact on true flash crowd users as they historically have been shown to abide by the normal usage characteristics of individual clients (i.e. requesting a few web pages). This is not to say that their collective actions represent normal traffic, just their individual request semantics. On the other hand, if an FRC attacker masquerades as a flash crowd, as seen in either the heavy-hitter or trace-driven attack patterns, then if the characteristics of this attack persist over a longer period of time, even if the individual clients rates are non-aggressive, the client usage patterns will be deemed to be anomalous and thus quelled.

At present, there does not exist a methodology that is able to distinguish increases in normal web activity from the actions of a FRC attack. In this regard, it is not sufficient to

attribute either an increase in aggregate request volume or increases individual request volume as malicious. The FRC detection methodology is volume independent in the sense that it focuses on the collective behaviors of clients and not the amount of requests that were enacted. This design principle is key in order to differentiate a surge in a web sites popularity versus fraudulent activity.

5.10 Future Work

The focus of this chapter, as was described in Introduction, was to detect FRC attacks from that of normal or increases in normal activity. Although positive results were achieved, there is still much work to be done in this problem space to truly reduce all FRC attack scenarios to that of nuisance activity. The detection mechanisms presented are for the broad analysis of request usage, and are not applicable for the identification of individual attack clients. Building on the results of this chapter, it would be ultimately desirable to perform the given detection methodology on a range of datasets that encompass and number of client volumes, website structure and purposes, and client demographics.

The detection scheme presented in this work was targeted at primary requests generated by clients. By analyzing only the primary requests, secondary requests were discarded before the training and testing phases of the methodology. Future research into exploring the same general approach but applied to the entire breadth of the web data logs would be interesting from the perspective of whether or not such analysis provides a more accurate profiling of normal client behavior and application of the presented detection metrics.

One of the primary objectives of this detection methodology was to find a solution that was independent of the aggregate request volume and that focused on the semantics of aggregate user behaviors. To develop a detection solution based on request volume, researching the feasibility of using the y-intercept of the regression line for the Zipf-like distribution would potentially provide an indication of fraud. Coupled with knowledge of the expect or normal request volumes, these metrics could provide a broad indication of the expect parameters for request volumes given a specific training window size.

As is evident in Figure 5.7, web usage follows a cyclical pattern dependent on the days of

the week. In the approach presented in this chapter, each day used in the training and test windows was done so without regard for which day in the week was present. If more days of data logs were available, it would be worth exploring if the usage characteristics for particular week days can be modeled in order to provide a more accurate profiling of daily aggregate web usage.

The attacks presented in this chapter enabled the attacker to have great latitude in his/her attack by having access to the training web logs before the onset of the attack. An beneficial research project would be to generate attack profiles, without knowledge of the training web logs, and based on the structure of a target website. This formulation of attack requests would require a live website and ability to collect the resultant logs. Such analysis and the attack patterns formalized would provide additional test cases utilizing the presented detection methodology against a diversity of new attacks.

5.11 Conclusion

As it is currently structured, the pay-as-you-go public cloud utility model is vulnerable to remote exploitation from any Internet-enabled device. Given the lack of attention and mitigation solutions for this open vulnerability, an attacker has the unrestricted ability to consume significant volumes of data usage that can result in a severe financial impact for a cloud consumer. The challenge this chapter sought to address was that of accurately differentiating aggregate data usage of legitimate clients from that of attack clients. Such a problem is difficult because requests only differ in the intentions of the attacker not in the structure or semantics of the request.

Presented in this chapter are three detection metrics that together provide the criteria capable of detecting three plausible FRC attack scenarios. To test the robustness of the proposed detection scheme against a worse-case scenario, the stated threat model enabled the attacker to access the log data before the onset of the a FRC attack. Despite the advantages of the attacker, the experimental results demonstrated that the proposed FRC detection scheme has had qualified success under very challenging attack scenarios.

CHAPTER 6. FRC ATTRIBUTION

Chapter contains modified content from the following submitted conference paper:

Idziorek, J., Tannian, M. and Jacobson, D. Attribution of Fraudulent Resource Consumption in the Cloud. *2012 IEEE 5th International Conference on Cloud Computing (CLOUD '12)*,

6.1 Abstract

Obligated by a utility pricing model, Internet-facing web resources hosted in the public cloud are vulnerable to Fraudulent Resource Consumption (FRC) attacks. Unlike an application-layer DDoS attack that consumes resources with the goal of disrupting short-term availability, a FRC attack is a considerably more subtle attack that instead seeks to disrupt the long-term financial viability of operating in the cloud by exploiting the utility pricing model over an extended time period. By fraudulently consuming web resources in sufficient volume (i.e. data transferred out of the cloud), an attacker (e.g. botnet) is able to incur significant fraudulent charges to the victim. This chapter proposes an attribution methodology to identify malicious clients participating in a FRC attack. Experimental results demonstrate that the presented methodology achieves qualified success against challenging attack scenarios.

6.2 Introduction

The offering of resources as a metered service is an essential characteristic that defines cloud computing [77]. Analogous to public utilities like electricity and gas, cloud consumers

are charged for computing resources like storage, processing, and bandwidth on a pay-per-use basis. As an example, consider a web-based service hosted in the cloud. Each GB of bandwidth consumed in support of client requests is applied to a utility pricing model and a fee is assessed to the cloud consumer. Pursuant to a Cloud Service Provider's (CSP) Terms of Agreement, cloud consumers are financially responsible for all bandwidth consumed in support of their web services whether clients consume these resources in good faith or not.

Obligated by a utility pricing model, public-facing web resources hosted in the cloud are vulnerable to Fraudulent Resource Consumption attacks. Unlike an application-layer DDoS attack that consumes resources with the goal of disrupting short-term availability, a FRC attack is a considerably more subtle attack that instead seeks to disrupt the long-term financial viability of operating in the cloud by exploiting the utility pricing model over an extended time period. By fraudulently consuming bandwidth in sufficient volume (i.e. data transferred out of the cloud), an attacker (e.g. botnet) is able to incur significant fraudulent charges to the victim. Such attacks are difficult to detect because the malicious clients' requests are non-aggressive, protocol compliant, and only different in the intent of the requester.

Building on the previously explored topic of FRC detection (Chapter 5), the main contribution of this chapter is an attribution methodology capable of identifying FRC attack clients. This work seeks to reduce the impact of fraudulent clients while minimizing the misclassification of legitimate clients. Different than application-layer DDoS solutions that focus on request rates of attack clients, the attribution methodology presented in this paper targets four aspects of client web browsing behavior. These four qualities are formulated into an attribution methodology that is tested through the use of three progressively challenging attack scenarios.

This chapter is organized as follows. Section 6.3 provides a brief risk analysis of utility computing. Related works are discussed in Section 6.4. Section 6.5 describes the datasets used for experimental purposes. The attribution methodology is presented in Section 6.6 and the experimental evaluation is provided in Section 6.7. Section 6.8 presents a more thorough analysis of FRC risk. A discussion and future work will be in Section 6.9 and lastly the conclusion is provided in Section 6.10.

6.3 Risk of Utility Computing

The utility pricing model is advantageous to a cloud consumer for it enables a low barrier of entry and avoidance of upfront capital expenses. In respect to bandwidth pricing, CSPs like Amazon Elastic Compute Cloud (EC2) [4], Microsoft Azure [122], and Rackspace [94] charge \$0.12/GB (up to 40 TB), \$0.12/GB and \$0.18/GB respectively for data transferred out of a cloud environment (i.e. bandwidth). While the utility model is convenient, it is not without its risks.

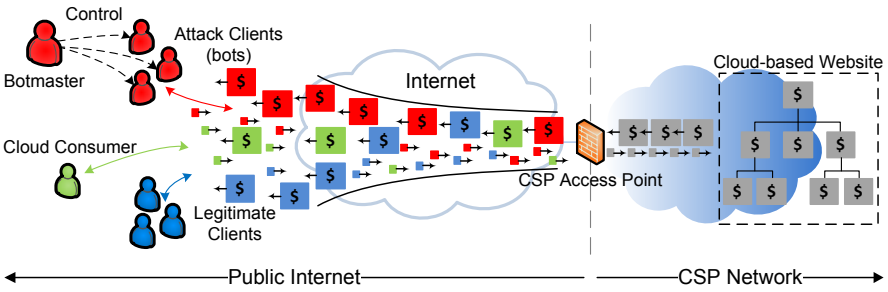


Figure 6.1: FRC Attack Illustration

As illustrated in Figure 6.1, each cloud-hosted web document represents a cost that is a function of the document's size and the CSP's pricing model. Thus each request serviced by a web site or web application translates to a direct cost that is assessed to the cloud consumer. Because malicious clients requests differ only in intent, all requests are serviced by the web application. The unlimited costs resulting from aggregate outbound replies triggered by malicious requests is the risk this paper seeks to help mitigate. This section will explore the *likelihood* that the utility pricing model vulnerability is exploited and the *impact* an FRC attack would have on a cloud consumer (i.e. victim).

The *likelihood* of a FRC attack is contingent upon the exploitability of the utility pricing vulnerability and the threat source's skill level, capacity, and motivation. Discovering the utility model vulnerability is a simple task for the attacker because data usage pricing is openly published [4, 94, 122] and the utility pricing model vulnerability has been discussed in both academic and non-academic forums [51]. Exploiting the utility model is simply a matter of making legitimate requests for publicly-hosted web content in sufficient volume to inflict a

malicious cost burden on the cloud consumer. As a result, a FRC attack could conceivably be performed by any Internet-connected device. Whether fraudulent requests originate from a simple Perl script or through the use of the more recently infamous Low Orbit Ion Cannon - an open-source software tool that enables individuals to perform or participate in DDoS attacks - there exists few technical barriers to conduct a FRC attack. The worst-case threat source, however, is that of a large botnet. Based on the sophistication and recent increase in number and volume of DDoS attacks [25], the worst-case threat source undeniably possesses the technical knowledge and resources necessary to conduct and sustain an impactful FRC attack.

Given the discoverability and ease of exploiting the utility pricing model vulnerability, the only significant factor preventing a FRC attack is the motivation of the attacker. Like botmasters that perform DDoS attacks, the source of a FRC attacker's motivation is wide-ranging and likely includes ego, hacktivism, monetary gain, extortion, revenge, creating a competitive advantage, and/or economic espionage [82].

The financial damage resulting from a FRC attack is largely a product of the attacker's average request intensity and the duration of the attack. While a FRC attack could easily exploit the utility model vulnerability with a DDoS attack, the attacker would do so at the increased risk of detection and ultimately mitigation. To enumerate the financial impact of an overt FRC attack, consider that the average bandwidth for a DDoS attack recorded in the last quarter of 2011 was 5.2Gbps [25]. Applied to the utility pricing model, at \$0.12/GB, if such an attack were sustained on the cloud model the resultant costs would be \$4.68/min, \$280.8/h, \$6739/day, and \$47,174/week. On the other end of the spectrum, an attacker could employ a slow-and-low attack strategy to avoid detection but would then be required to sustain the attack over a longer duration of time to achieve the desired cost impact. Given a more subtle attack approach, all that is necessary for a 100,000 node botnet to double the data usage costs of a website that averages five requests per second would be for each bot client to request eight web pages a day. Such a un-assuming attack is the focus of this work as it would go undetected by current firewalls, DDoS mitigation, and intrusion/detection solutions.

6.4 Related Work

The closest body of work to that of FRC attribution is research that seeks to identify bot clients participating in application-layer DDoS attacks. Given that both attacks utilize HTTP requests to accomplish their goals, these works have served as a natural starting point for exploring FRC attribution options. The following analysis is presented to motivate the need for attribution methodologies that are able to detect much less intense attacks.

To identify application-layer DDoS attack clients, Ranjan et al. [97] devised statistical methods to build profiles of normal client behaviors by considering client inter-session arrival times, inter-request arrivals times and session workload parameters. Similarly, Oikonomou et al. [87] examined a number of web browsing features to distinguish bot clients that exhibit aggressive behaviors. Central to both of these attribution methodologies is the reliance on time-based parameters (i.e. request rates). While these researchers report success against application-layer DDoS attacks, such methodologies would prove ineffective against a FRC attack because individual clients need not resort to short-term aggressive request patterns in order to accomplish the attacker's long-term goals.

To differentiate legitimate clients from bot clients, Kandula et al. [58] proposed an attribution scheme based on graphical puzzles. While effective in some contexts, such an approach is not generally considered a practical solution [87], especially as a proactive approach for web content that is intended to be publicly hosted. More transparent approaches include the use of honey-tokens [40] - invisible decoy hyperlinks errantly requested by automated bots. Although clever, this attribution methodology only considers a limited threat model.

Other approaches to differentiating legitimate clients from DDoS bots includes analyzing the transitional probabilities between successive client requests [87, 123, 124]. The premise of these works is that bots are unable to deduce common request sequences for a particular web site. While potentially applicable to FRC attribution, these solutions only consider a single aspect of a client's overall request characteristics and thus are restricted to a limited threat model not considered sufficiently general for this work. Additionally, Kruegel et al. [68] proposed a web-based anomaly detection methodology for the identification of attacks that attempted to

exploit web-based software or configuration vulnerabilities. This work is distinctly different in that the objectives of a FRC attack is not application or host compromise, but to use the utility pricing model as a weapon. In this respect, the proposed attribution methodology focuses on anomalous usage characteristics of individual clients, not on the semantics of a query string in a HTTP GET request.

More practical and less academic solutions include the assistance of web server anti-DDoS software modules. Developed for the Apache web server, *mod_evasive* is a DDoS detection and mitigation tool that prohibits individual IP addresses from requesting the same webpage more than three times in a second and issuing more than 50 concurrent requests per second [129]. Through statistical tracking of IP addresses and URIs, mod_evasive also perform blacklisting of offending clients. While effective for DDoS attacks, this solution directly targets the aggressive rate of requesting clients, which is not considered a quality of a FRC attack client considered in this work.

From this survey of related works, it should become apparent that current application-layer DDoS attribution methodologies are not tailored to identify FRC attack clients. Instead of focusing on time-based parameters, which often go hand-in-hand with resource exhaustion attacks, attribution solutions that explore the behavioral characteristics of malicious client requests are instead needed to identify surreptitious FRC attack clients that attempt to mimic legitimate client behaviors.

6.5 Dataset Description

6.5.1 Web Log Properties

A web server log is a time-based record of all clients' content requests and provides the necessary observations for extracting features for deriving profiles of normal web activity. Figure 6.2 provides an illustration of the web usage features used for the purposes of attribution in this work and also provides the context for the data preprocessing steps.

Each vertical line in Figure 6.2 depicts individual web requests that compose a web server log. Denoted as tall lines, each primary web document request is an explicit request from

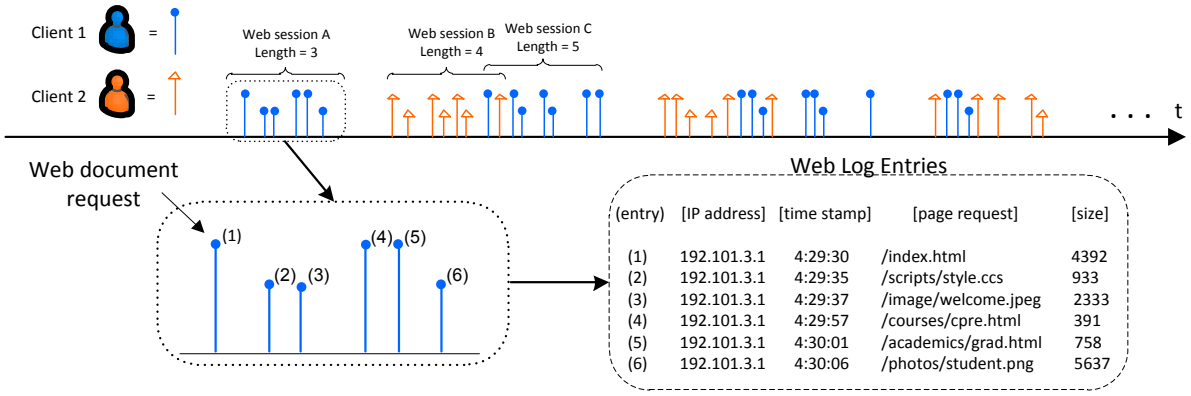


Figure 6.2: Web Log Components

a client. As is typical for web documents, a single request often initiates other secondary in-line requests to retrieve other web content embedded within the primary web document such as figures, scripts or videos. Secondary in-line requests, as seen in the callout in Figure 6.2, are depicted as shorter lines that follow a primary request. For example, the primary request of (1)/*index.html* is followed by two secondary requests (2)/*scripts/style.css* and (3)/*images/welcome.png*. While both primary and secondary requests are reflected in a web log, without the live website from which a web log was obtained, it is difficult to differentiate primary and secondary requests with complete accuracy. Client-side and distributed network caching further complicate the task of reconstructing these relationships from a web log. Therefore, in lieu of this ambiguity, web documents reflected as URLs for HTML documents or URIs were assumed to be primary requests and other web document requests such as pictures and styles sheets were discarded during the preprocessing of the log as they are not necessary to profile normal client behaviors.

Several terms need to be defined before discussing the proposed attribution methodology. They are:

Request Volume - The quantity of primary requests invoked by a client within an observation time period.

Web Session - A set of consecutive primary requests generated by an individual client during a single viewing period is known as a web session. As seen in Figure 6.2, *web session A* contains three primary web requests and thus has a web session length of three. Although it

is logical to think of usage bursts as sessions, web protocols do not inherently support such a distinction. In order to construct a session, a pause in a client's recorded activity longer than 900 seconds is considered to be a separation between sessions [67].

Session Length - The number of primary requests in a web session.

Average Session Length - An individual client may invoke one or more web sessions of varying lengths and thus this term denotes the mean of these lengths.

Session Volume - The quantity of web sessions attributed to a single client within an observation period.

6.5.2 Experimental Datasets

Two datasets consisting of 56 consecutive days of web logs from two websites were partitioned into 28-day datasets for training and test. The first dataset (denoted NASA) is a historically popular research dataset from a busy NASA web server [83]. The second dataset (denoted ECpE) originates from our department's public web server. It is assumed that these weblogs are a reasonable representation of each of the respective sites beyond the 56 days collected. Table 6.1 provides a description of these datasets in respect to the client characteristics. These descriptions will prove to be useful when evaluating the experimental results. Although the purpose, volume of requests, and nature of the websites are quite different, there are consistencies between the websites that enable the formulation and experimental testing of a generalized solution that is tuned but not designed for a specific web site.

6.6 Attribution Methodology

Attribution in the context of this work is the ability to accurately identify malicious clients - based on IP addresses - from that of legitimate clients. The objective of this section is to describe the anomaly detection methodology evaluated in Section 6.7.4. This attribution methodology exploits the choices that an attack client must make under the hypothesis that it is difficult to replicate normal client behavior without having access to web server logs or the training dataset.

Table 6.1: Description of Experimental Datasets (all measures cover a 28-day observation period)

Metric	ECpE	NASA
Total Requests		
Training	45 953	564 090
Test	45 268	413 839
Avg Reqs per Client		
Training	5.54	7.18
Test	5.64	5.99
Avg Session Length		
Training	2.26	2.12
Test	2.37	2.00
Avg Sessions per Client		
Training	2.93	3.17
Test	2.98	2.97

This methodology seeks to find statistical outliers among clients based on four observed usage characteristics: request volume (req_{vol}), average session length (ses_{avg}), volume of sessions (ses_{vol}), and chi-square statistic (chi_{stat}) of requested documents. In the evaluation of a client, each attribution variable returns a probability score between 0.0 and 1.0 indicating the likelihood that the resource usage footprint for a client is anomalous. As the attribution variable approaches 1.0, the likelihood of usage behavior being anomalous increases. The four respective metrics are summed together to produce an overall *attribution score* (ATT_{SCORE}) for each client.

$$ATT_{SCORE} = req_{vol} + ses_{vol} + ses_{avg} + chi_{stat} \tag{6.1}$$

After the ATT_{SCORE} is calculated for a client, it is then compared to a threshold that is the cutoff between designating a client as benign or malicious. This threshold is the sensitivity parameter of the attribution methodology. The following subsections describe each of the respective attribution metrics in detail.

6.6.1 Request Volume

Based on the objective of a FRC attack, it is in the best interest of an attack client to consume as much bandwidth as possible over a certain duration of time without being identified. As a result, the per client request volume becomes a natural discriminator of a client's website usage, and is therefore useful in performing anomaly detection. Figure 6.3 shows the cumulative distribution function (CDF) for the per client request volumes for both the ECpE and NASA training datasets over a period of 28 days. Although the two datasets are an order of magnitude apart in terms of total request volume, the per client request volumes seen in these datasets exhibit very similar characteristics, an observation that is repeated for each of the remaining attribution metrics. Due to the overwhelming majority of clients that invoke very few requests (Table 6.1), the CDF describing per client request volumes was constructed with a minimum threshold of five requests to provide a larger contrast in the distribution of clients that invoke the most requests.

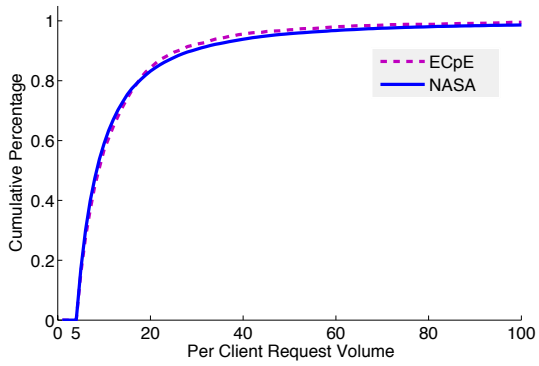


Figure 6.3: Request Volume CDF (min. threshold of 5 requests)

Based on the CDF in Figure 6.3, each client in the test dataset is assessed an attribution variable value, $req_{vol} = F_X(x)$, between 0.0 and 1.0 by determining the probability that a given normal client's request volume x would be less than or equal to that which was observed overall such that:

$$F_X(x) = P(X \leq x) \tag{6.2}$$

Thus the more requests that a particular client makes, the larger the req_{vol} value will be

assigned to that particular client. As seen in Figure 6.3, a client requesting more than 20 requests over 28 days is in the 80^{th} percentile resulting in a req_{vol} score greater than 0.8.

6.6.2 Session Volume

In an attempt to avoid discovery and to mimic that of normal client usage patterns, an individual attack client participating in a FRC attack is likely to distribute their resource consumption over the course of many days by launching multiple fraudulent web sessions. Without access to the web server log, the attacker does not know the amount of web sessions a normal client typically makes over a given time period. Figure 6.4 depicts the CDF of session counts per individual clients for both the ECpE and NASA training datasets. Similar to the rational for the request volume CDF, a minimum threshold of two web sessions is considered.

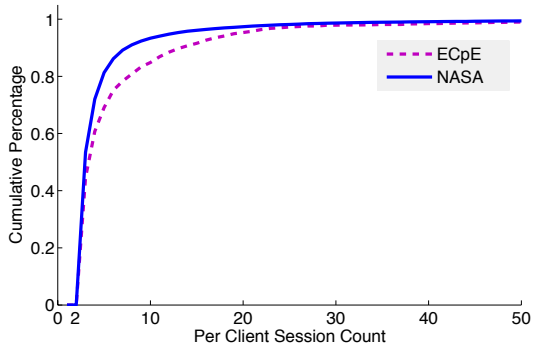


Figure 6.4: Sessions per Client CDF (min. threshold of 2 sessions)

The ses_{vol} attribution variable is assigned a probability value between 0.0 and 1.0 resulting from applying Equation 6.2 within the context of the session count CDF as shown in Figure 6.4. The probability value returned is the likelihood a normal client would have launched a less than or equal number of sessions than was invoked by the overall client population (i.e. n^{th} percentile). Given that the per client average web session is nearly three for both datasets (Table 6.1), malicious clients requesting slightly more than three sessions over a 28-day period will likely be flagged as anomalous.

6.6.3 Average Session Length

In addition to distributing requests over multiple web sessions, an attacker attempting to mimic normal behavior must also determine each web session length. As shown in the CDF for the per client average session length, well over 80% of individual clients exhibit an average session length less than that of five requests.

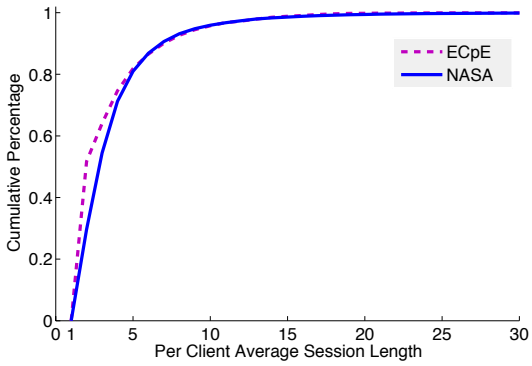


Figure 6.5: Average Session Length CDF

The ses_{avg} score is the probability that a normal client's average session length is less than or equal to that of what was observed overall. In order for an attacker to avoid a high ses_{avg} score, the attacker is forced to initiate more web sessions, which in turn contributes to a higher ses_{vol} score. Like the other attribution variables, ses_{vol} will take on values from 0.0 to 1.0.

6.6.4 Chi-Square Statistic

Pearson's chi-square test has been used as an anomaly detection methodology to compare multinomial distributions in a number of related contexts including network-layer DDoS detection [34] and intrusion detection [68]. In the context of the FRC attack, however, Pearson's chi-square statistic is instead used to evaluate the actual web pages a client requests in comparison to the web page request frequency distribution for an entire website. The Zipf-like distribution [132] for the ECpE training dataset is shown in Figure 6.6 - a log-log plot of document frequency (i.e. popularity) vs. document rank. Used in a similar context for FRC detection(Chapter 5), the Zipf-like distribution broadly states that 10% of the requested documents are requested 90% of the time. Discrete bins representing continuous ranks of requested

web documents can be formed based on what the Zipf-like distribution states. As shown in Figure 6.6, two bins are formed. Bin 1 contains the m most popular documents and Bin 2 contains the remaining documents in the web log. When comparing a client's collection of web document requests to overall website usage, these bins form probabilities that indicate, given k requests, X% of the requests are expected to fall into Bin 1 and likewise Y% of the requests fall into Bin 2. This test leverages the notion that a significant fraction of normal client behavior is reasonably self-similar to the overall client population.

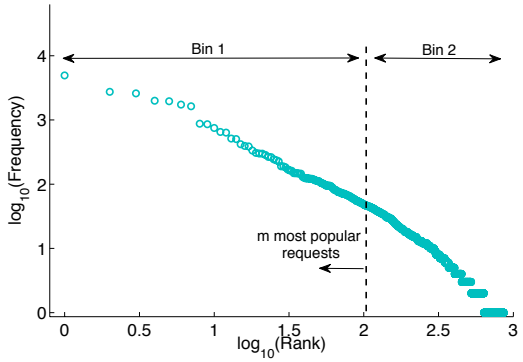


Figure 6.6: Application of Chi-Square Test to a Zipf Distribution

Given these preliminaries and given a client's collection of ranked web requests, let the ranked web pages be grouped into two or more discrete bins $B_1, B_2, ...B_k$ and let each bin represent a probability $\pi_1, \pi_2, ...\pi_k$ such that given n observed web requests, the expectation for each bin is $E_i = n\pi_i$. Partitioning a client's requests into k bins, n_i is defined as the number of requests that fall in the i^{th} bin and E_i is the expected number of requests based on observations from the training dataset. Under the premise that the frequency distribution established from the training dataset is consistent with a test client's requests, the chi-square statistic can be calculated as follows:

$$\chi^2 = \sum_{i=1}^{B} \frac{(n_i - E_i)^2}{E_i} \tag{6.3}$$

While often used for hypothesis testing, the chi-square statistic in this context is instead used as a relative measure of similarity or dissimilarity between individual client request distributions and the overall population distribution. For each client in the training dataset, a chi-square

statistic is computed and an overall CDF, as shown in Figure 6.7, is constructed. Based on this CDF, Equation 6.2 can be applied to assign a value (0.0-1.0) to chi_{stat}. While it would be tempting to assign a p-value to chi_{stat} based on the chi-square statistic, the application of a p-value in this context would be erroneous.

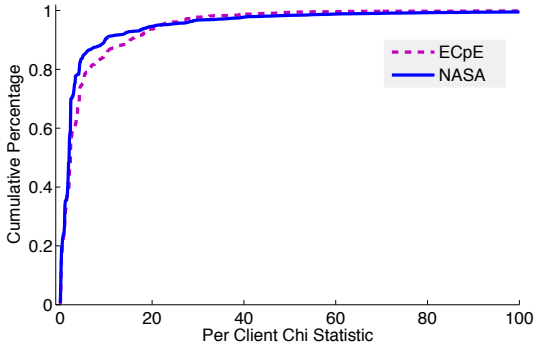


Figure 6.7: Chi-Square Statistic CDF

The power of this attribution variable is that it is difficult for an attacker to prescribe a page request frequency distribution that resembles that of which was observed in the training dataset. Although this particular attribution variable is subject to high false-positive rates, the overall methodology operates under the assumption that legitimate clients that exhibit a particularly high chi_{stat} will achieve typical statistics for the other variables and thus not be classified as malicious.

The overall objective of this attribution methodology is to limit the amount of resources that an attacker can consume without being detected. As presented, it is difficult for a FRC attacker to consume a large quantity of resources without drawing suspicion. Although the range of typical usage characteristics appear to be tightly bounded, there are legitimate clients that deviate from normal and the experimental evaluation of the proposed scheme will uncover the inevitable accuracy trade-offs of attacker attribution.

6.7 Experimental Evaluation

In order to conduct a FRC attack, the attacker must devise a methodology for individual attack clients by specifying: how many web documents, which specific documents, how many

sessions, and the length of the sessions an individual bot node will make. This section focuses on the tactical decisions an attacker must consider in order to conduct a FRC attack. Some of the decision combinations have been instantiated for testing and the corresponding experimental results are presented. For the sake of brevity, not every possible combination of attack parameters can be evaluated and presented here. Instead, a few specific FRC attack profiles have been crafted in order to enable the evaluation of the effectiveness of the attribution methodology. Although attribution criteria is applied to a 28-day test dataset, the methodology is not confined to this observation period. Instead, the methodology is sensitive to the usage characteristics of clients' requests.

6.7.1 Attacker Simulation

Three progressively challenging attack scenarios have been chosen to illustrate the effectiveness and limits of the presented methodology. Although in practice it would be safe to assume that the attacker does not have access to a web server's logs (i.e. the training dataset) prior to attacking, for two of the three attack scenarios evaluated this assumption is relaxed to give the attacker an advantage in order to challenge the proposed methodology. In each of the attack scenarios the attacker is utilizing a 100,000 node botnet. Each attack is implemented by synthesizing and interweaving attack entries into the test datasets. The attribution methodology is then applied to the test datasets after having trained on the training datasets.

6.7.1.1 Random Attack

The most naïve of the attack methodologies considered is where an attacker randomly requests web pages from a website without considering the popularity or the contextual relation between consecutive requests. Similar to spidering a website, an attacker requests a webpage and then randomly requests one of the hyperlinks present in the retrieved webpage. This process is repeated until a web session reaches the specified session length. In this scenario, the attacker randomly makes between five and ten random content requests per web session.

6.7.1.2 Prescribed-Sessions

Under the assumption that the attacker knows the top-100 most requested web sessions of at least three documents in length, the prescribed-sessions methodology simulates a more sophisticated attack in which each attack client is tasked with simply replaying the most popular web sessions experienced by the website under attack. In this case, the attacker does not determine the session length directly, but instead by the observed length of the chosen prescribed session.

6.7.1.3 Trace-Driven

The trace-driven attack methodology is the most challenging attack scenario for the defender and is considered the worst-case scenario. By replaying web sessions as captured in the training dataset, the attacker is able to craft an attack that is derived from how legitimate clients actually use a particular website. Like the prescribed-session methodology, each web session length is determined by the observed session chosen by the attack client.

6.7.2 Evaluation Criteria

Given the attribution metrics described in Section 6.6 and the evaluation of an individual client, there are four possible classification outcomes. If a client is malicious and is classified as positive, it is denoted as a true positive (TP); if a client is legitimate and is classified as negative, it is denoted as a true negative (TN); if a client is malicious but is classified as a negative, it is denoted as a false negative (FN); lastly, if a client is legitimate but is classified as a positive, it is denoted as a false positive (FP). In order to evaluate the attribution scheme, a False Positive Rate (FPR) and a False Negative Rate (FNR) is computed as a percentage for each test iteration.

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN} \tag{6.4}$$

$$FalseNegativeRate(FNR) = \frac{FN}{FN + TP} \tag{6.5}$$

6.7.3 Baseline FPR

When analyzing malicious bots, a natural starting point would be to examine the total volume of requests invoked by each client and use this threshold as a identification criterion. For instance, if any client invoked more than 40 documents, regardless of their other usage characteristics, the client would be considered malicious and be challenged by some form of authentication. As a baseline measure of comparison, if the request volume were indeed the sole discriminant (i.e. Request Threshold), then Table 6.2 displays the FPRs for the ECpE and NASA datasets. It follows that the FNRs for each these measures would subsequently be 0.00% if the attacker requested greater than or equal to the volume of the set request threshold. A goal of this attribution methodology is to significant improve upon these FPRs while maintaining FNRs that relegate aggregate requests to that of nuisance activity. Depending on the size of an attacking botnet and the set request threshold of the attribution methodology, a FNR of 0.00% or nearly 0.00% is not necessary if the data usage from the malicious requests that register as FNs do not eclipse J1 in Figure 5.6a. While a FNR of 0.00% is ultimately desirable, the following experimental results will be analyzed in the context of minimizing the FPRs while maintaining tolerable FNRs.

Ultimately the goal of a FRC attacker is to achieve a cost burden onto the cloud consumer that equates to a certain number of terabytes of data. While it would be logical to model the data usage footprint of each individual client to look for statistical outliers, as is also discussed in the future work, there does not exist a current body of work to support this analysis. To model and simulate individual clients' data usage footprints, one must be able to accurately model the dependencies between primary and secondary requests for a given website. Such modeling can only be done with access to the live website from which the archived log entries were originated. Additionally, the caching of both primary secondary requests must also be taken into account and historical logs provide incomplete evidence to make accurate assertions. Despite this need for future work and given the presented attribution methodology, the attacker is still constrained to generating attack behaviors that are in accordance with normal client behaviors. Based on these constraints, attack optimization in terms of data usage may prove

Table 6.2: ECpE and NASA Request Threshold FPRs

Request Threshold	ECpE				NASA		
	FP	TN	FPR		FP	TN	FPR
0	8007	0	100.00		67008	0	100.00
5	2134	5873	26.65		15818	51190	23.61
10	1128	6879	14.09		7277	59731	10.86
15	622	7385	7.77		4283	62725	6.39
20	383	7624	4.78		2905	64103	4.34
25	249	7758	3.11		2113	64895	3.15
30	179	7828	2.24		1626	65382	2.43
35	137	7870	1.71		1311	65697	1.96
40	108	7899	1.35		1044	65964	1.56
45	96	7911	1.20		853	66155	1.27
50	77	7930	0.96		701	66307	1.05
55	60	7947	0.75		609	66399	0.91
60	52	7955	0.65		538	66470	0.80
65	48	7959	0.60		473	66535	0.71
70	46	7961	0.57		414	66594	0.62
75	42	7965	0.52		375	66633	0.56
80	40	7967	0.52		338	66670	0.50
85	33	7974	0.41		309	66699	0.46
90	28	7979	0.35		276	66732	0.41
95	25	7982	0.31		249	66759	0.37
100	20	7987	0.25		232	66776	0.35

to be difficult for the attacker.

6.7.4 Experimental Results

Tables consolidating the FPRs and FNRs for each attack scenario and dataset follow shortly. Each of these tables has the same structure. The first column denotes the threshold applied to each computed Att_{SCORE} resulting in client attribution. The scale of 3.00 to 3.45 was chosen to capture a gradient of FNRs in contrast to the FPRs, which are presented in the second column. The remaining columns are each labeled by the number of requests the attacker made and the cells populated underneath these column headings represent the resultant FNRs for a given threshold specified in the first column. Each FNR cell value is an average of three distinct attack simulation runs using the same attack parameters.

Due to the many ways a given website can be utilized, the Att_{SCORE} for legitimate clients yields a considerable amount of variance leading to undesirable FPs. Unlike an intrusion detection system in which a FN could potentially result in an undetected compromise of a system, a FN in the context of a FRC attack is comparatively less worrisome. The result of a misclassified attack client is incremental charges to the cloud consumer for those resources the attack client consumed. Because of this fundamental characteristic of a FRC attack, Section 6.8 will provide context for the FNRs through a risk analysis.

6.7.4.1 Random Attack Results

The results reported in Tables 6.3 and 6.4 are for that of attack clients requesting random pages and issuing web sessions of random length between 3 and 13 documents. As can been, when employing such an attack strategy, the individual attack clients have a difficult time requesting more than 30 requests without being flagged as malicious. The motivation for choosing a random session lengths of the stated values was derived from attack scenario crafted by Oikonomou et al. for a similar profiling of attack clients behaving in a DDoS attack [87]. In this work, attack clients' behavior was modeled as an attacker requesting between 5 and 50 documents per session. Using a common dataset (i.e. NASA) and based on the results of the CDF in Figure 6.5, such a model for session lengths is clearing going to register as anomalous. Instead, this attack scenario considers attack strategy more aligned with that of a realistic, but naive attacker. The objective is to show that attack clients cannot employ a primitive attack strategy and achieve success against the presented attribution methodology.

Based on the anomalous nature of the attack scenario, if a FPR of approximately 1.43% is acceptable to the cloud consumer, the threshold can be set to identify malicious clients requesting 25 or more web documents without registering a single FN. As is often the case in the evaluation of anomaly detection systems, one thing to observe is that as the threshold increases (i.e. sensitivity decreases) so too does the FNRs while the FPRs decrease. This observation holds true for the remaining attack scenarios and context for this inevitable trade-off will be discussed in Section 6.8. Moreover, the FPRs are the same for a given website across attack scenarios because the attribution methodology and original log entries remain constant.

Table 6.3: Random - ECpE Attack Results (%)

		FNR (Reqs per Attacker)									
Threshold	FPR	20	25	30	35	40	45	50	55	60	65
3.00	2.77	14.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.05	2.50	14.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.10	2.16	14.57	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.15	1.77	15.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.20	1.43	71.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.25	1.20	71.20	0.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.30	1.02	97.50	36.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.35	0.84	100.0	36.43	8.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.40	0.52	100.0	83.87	8.37	0.33	0.00	0.00	0.00	0.00	0.00	0.00
3.45	0.39	100.0	100.0	54.90	0.33	0.00	0.00	0.00	0.00	0.00	0.00

To put the result in Table 6.3 into perspective, consider the shaded cell denoted by the superscript one. If using the per-client request volume were the sole attribution criterion, then at an attack intensity of 30 requests per attacker, based on the results in Table 6.2, the cloud consumer could achieve a FNR = 0.00% if a FPR = 2.24% were acceptable. Given that the objective of the FRC attribution methodology is not to detect every malicious client but instead a sufficient amount of malicious clients in order to reduce the impact of a FRC attack, then a FNR = 8.37% would be preferable given the comparably less impactful FPR = 0.84% yielded by the given attribution methodology.

It is highly tempting to compare attribution performance across websites in addition to across attack scenarios. However any conclusions drawn would be erroneous. The websites are sufficiently dissimilar resulting in "apples to oranges" comparisons. With that in mind, the explanation for lower FNRs in the NASA results is due to the sessions per client CDF for the NASA dataset resulting from a higher volume at lower sessions counts. The NASA dataset yields a slightly higher ses_{vol} for the same attack parameters and thus produces a consistently lower FNR.

Applying Equation 6.1 to the described attack scenario for the NASA dataset illustrates why this particular attack scenario fails for the FRC attacker. As highlighted in Table 6.4, consider the attack scenario in which attack clients request 20 web documents. First, because

Table 6.4: Random - NASA Attack Results (%)

		FNR (Reqs per Attacker)									
Threshold	FPR	20	25	30	35	40	45	50	55	60	65
3.00	2.22	15.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.05	1.99	15.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.10	1.78	15.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.15	1.59	15.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.20	1.42	15.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.25	1.25	16.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.30	1.12	72.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.35	0.99	72.90	35.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.40	0.84	100.0	35.73	7.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.45	0.73	100.0	35.77	7.97	0.50	0.00	0.00	0.00	0.00	0.00	0.00

the attacker is simply requesting random documents, this usage pattern is not indicative of the overall client population. As a result, attackers consistently score a chi_{stat} with $\mu = 1.0$ and $\sigma = 0.0$. While enacting sessions lengths between 3 and 13 documents limits the ses_{avg} score to $\mu = 0.6$, it does provide a some amount of variability leading to a $\sigma = 0.05$. Even at 20 requests, this attack scenario yields a req_{vol} score of $\mu = 0.80$ and a ses_{vol} score of $\mu = 0.51$ with $\sigma = 0.23$. It is this variability that is responsible for the changes in the FNR for 20 requests (i.e., FNRs = 15.13, 16.77, 72.73, 72.90, and 100.0). While the chi_{stat} and ses_{avg} scores remain fairly consistent as the request intensity increases, both the req_{vol} and ses_{vol} do increase as the request intensity increases and thus the FNR decreases given the common thresholds. These results make it difficult for the attacker to request much more than 25 requests per bot when simply requesting random web pages.

6.7.4.2 Prescribed-session Attack Results

In this attack strategy, the attacker is given knowledge of the 100 most requested web sessions from the training dataset of length three or greater. It is conceivable that an attacker would attempt to guess a number of common web sessions and distribute these session among a botnet. Despite the advantage of being privy to such knowledge, as seen in Tables 6.5 and 6.6, individual attack clients yield limited success in this attack scenario. From the attacker's

perspective, the achieved results are slightly better than the random attack scenario, because the prescribed-pattern attack benefits from a lower ses_{avg} score produced by replaying actual web sessions.

Table 6.5: Prescribed-Session - ECpE Attack Results (%)

		FNR (Reqs per Attacker)									
Threshold	FPR	20	25	30	35	40	45	50	55	60	65
3.00	2.77	12.80	0.50	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.05	2.50	46.27	3.33	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.10	2.16	57.83	3.40	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.15	1.77	100.0	3.40	0.97	0.03	0.00	0.00	0.00	0.00	0.00	0.00
3.20	1.43	100.0	20.00	0.97	0.03	0.03	0.00	0.00	0.00	0.00	0.00
3.25	1.20	100.0	59.20	5.70	0.13	0.03	0.00	0.00	0.00	0.00	0.00
3.30	1.02	100.0	100.0	58.93	0.33	0.30	0.00	0.00	0.00	0.00	0.00
3.35	0.84	100.0	100.0	99.90	15.57	1.93	0.03	0.00	0.00	0.00	0.00
3.40	0.52	100.0	100.0	100.0	86.40	12.93	0.57	0.07	0.00	0.00	0.00
3.45	0.39	100.0	100.0	100.0	100.0	93.60	22.47	1.60	0.27	0.00	0.00

One of the primary reasons the prescribed-session attack is not able to overcome the attribution methodology is that despite knowledge of popularly requested web sessions, individual attack clients are not able to decrease their chi_{stat} score, which, for an attack intensity of 20 requests, has a $\mu = 0.98$ and $\sigma = 0.03$. This example plays to the strength of the chi_{stat} attribution metric. Because the individual documents that compose the 100 most popular web sessions primarily fall into Bin 1 in Figure 6.6, the expectation that roughly 10% of the request generated by the attacker should fall into Bin 2 is seldom fulfilled and thus the consistently high chi_{stat} scores are recorded leading to a higher overall ATT_{SCORE}.

Of the 100 web sessions chosen by the attacker for the ECpE dataset, the majority of the sessions are of length three or four. Generating web session lengths closer to that experienced by normal users reduces the attack clients ses_{avg} score and significantly reduces the respective standard deviation. While advantageous for the attacker, the effect this reduction has the ses_{avg} increases the ses_{vol} score to a $\mu = 0.75$ with $\sigma = 0.02$ at a request intensity of 20 documents - a trend that continues increase and request intensity increases.

As can be observed for each of the attack scenarios, the significant increases in FNRs

between thresholds and attack request volumes can be attributed to the homogeneity of the attack profiles or, in other words, a lack of variance in the attribution metric scores a result to the attacker scenarios. As a result, an increase in the threshold value of 0.05 can yield noticeable differences or no difference at all in the FNRs. This point can be illustrated in Table 6.5 between the two highlighted cells.

Table 6.6: Prescribed-Session - NASA Attack Results (%)

		FNR (Reqs per Attacker)									
Threshold	FPR	20	25	30	35	40	45	50	55	60	65
3.00	2.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.05	1.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.10	1.78	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.15	1.59	91.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.20	1.42	100.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.25	1.25	100.0	51.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.30	1.12	100.0	99.93	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.35	0.99	100.0	100.0	100.0	77.13	0.00	0.00	0.00	0.00	0.00	0.00
3.40	0.84	100.0	100.0	100.0	99.97	35.70	0.00	63.23	0.00	0.00	0.00
3.45	0.73	100.0	100.0	100.0	100.0	100.0	99.97	63.23	22.80	0.00	49.43

The CDFs in Figures 6.3 and 6.4 show that given the same attack parameters, the attribution scheme would yield similar results with the exception of the session count CDF (Figure 6.5). The attack parameters derived from the respective training datasets are dissimilar enough to produce a noticeable contrast in final FNR results. On average, the replayed web sessions for the NASA conformed less closely to that of normal traffic and thus exhibit higher ATT_{SCORE} on average than that of the 100 most popular web sessions observed in the ECpE dataset.

6.7.4.3 Trace-Driven Attack Results

A trace-driven attack is extremely challenging for anomaly detection because the attacker has managed to emulate normal usage behavior. In order to identify significant variation in FNR performance, the range of per client request volume for this attack scenario has been extended. Increasing the per request volume translates to each client launching more trace driven sessions thus providing a greater opportunity for detection. As shown in Tables 6.7

and 6.8, even though an attacker is requesting a significant amount of web documents (i.e. req_{vol}) and thus web sessions (i.e. ses_{vol}), the overall FNRs remain quite high. As the number of requests and web sessions increase, the attacker approaches the 99^{th} percentile for each dataset's request volume CDF and session count CDF resulting in req_{vol} and ses_{vol} scores returning values of where $\mu = 0.98$ is $\sigma = 0.03$. However, because the attacker is simply replaying web sequences of normal request patterns for a website, both the chi_{stat} ($\mu = 0.62$ and $\sigma = 0.31$) and ses_{avg} ($\mu = 0.44$ and $\sigma = 0.24$) scores remain within the bounds of normal despite the increase in request volume and thus do not contribute sufficiently to the overall score ATT_{SCORE} necessary to yield lower FNRs.

Table 6.7: Trace-Driven - ECpE Attack Results (%)

		FNR (Reqs per Attacker)									
Threshold	FPR	20	40	60	80	100	120	140	160	180	200
3.00	2.77	90.47	69.57	64.90	61.73	63.07	59.67	59.30	59.03	57.53	58.80
3.05	2.50	93.97	73.43	67.53	63.93	67.10	63.57	62.87	63.73	60.40	62.43
3.10	2.16	96.50	77.37	70.77	67.03	70.33	66.37	65.17	66.60	62.30	64.70
3.15	1.77	98.50	81.90	75.87	72.17	72.97	69.40	68.87	68.40	64.87	66.90
3.20	1.43	99.37	85.37	78.50	76.50	77.17	73.83	73.70	74.03	70.67	70.87
3.25	1.20	99.83	87.97	81.47	79.30	80.03	76.90	77.50	78.73	74.67	75.30
3.30	1.02	100.0	91.33	84.80	82.60	83.10	80.20	80.03	81.57	77.40	77.90
3.35	0.84	100.0	93.90	87.57	86.23	87.20	84.33	83.50	84.10	81.23	82.17
3.40	0.52	100.0	96.20	91.53	90.53	90.83	89.17	88.60	88.87	86.80	87.50
3.45	0.39	100.0	97.67	94.77	93.57	94.53	93.70	93.87	93.37	92.60	92.37

Similar to the prescribed-path scenario, the differences in FNRs are mainly a product of the random web session replayed by the attack clients. Although high, the observed FNRs were anticipated. If an attacker is able to learn the web session patterns of normal clients and use this knowledge to mount an attack, then the resulting attack client request footprints should largely be within the scope of normal. Under more formidable attack patterns like the trace-driven scenario, Equation 6.1 can be altered by assigning a weight to each variable placing greater emphasis on client's usage characteristics that are more likely to be anomalous (e.g., req_{vol} or ses_{vol}) for a given FRC attack.

In many ways, enabling the attacker to have access to the training logs is a hefty assumption.

Table 6.8: Trace-Driven - NASA Attack Results (%)

		FNR (Reqs per Attacker)									
Threshold	FPR	20	40	60	80	100	120	140	160	180	200
3.00	2.22	70.10	50.60	44.77	42.90	39.50	39.63	37.73	39.00	35.73	37.07
3.05	1.99	79.90	54.90	47.77	44.73	42.30	41.57	40.87	41.40	39.43	40.33
3.10	1.78	80.63	57.87	50.40	48.77	44.00	45.03	42.40	44.37	41.37	42.57
3.15	1.59	89.50	62.70	53.40	50.87	45.47	46.10	44.07	45.77	43.17	44.60
3.20	1.42	94.60	67.07	57.47	54.30	48.07	48.40	46.07	48.23	44.37	46.07
3.25	1.25	97.00	73.97	61.70	58.37	51.90	51.13	48.97	50.67	47.00	49.47
3.30	1.12	98.90	79.50	67.17	64.43	57.07	55.03	52.67	53.50	49.37	52.63
3.35	0.99	99.70	83.67	74.87	69.67	62.33	60.67	58.37	59.43	55.97	56.67
3.40	0.84	100.0	86.90	78.80	77.70	70.63	68.63	66.87	66.23	64.47	64.20
3.45	0.73	100.0	90.70	84.23	82.33	76.87	75.60	74.10	73.30	71.03	71.60

Even though the attribution methodology yields undesirable FNRs for large volumes of requests, this behavior was expected as the attribution methodology was designed for this very purpose. An objective of the attribution methodology was to decrease the FPRs for legitimate clients that request a larger than average amount of web documents. Instead of issuing an attribution threshold based solely on request volume, as is shown in Table 6.2, the attribution methodology operates on four criterion that together describe normal traffic. This way, a legitimate client that requests more than 40 documents and subsequently registers higher than normal scores for the req_{vol} or ses_{vol} metrics can compensate by scoring closer to normal for the ses_{avg} and chi_{stat} scores.

6.8 FRC Risk Analysis

Exploring the proposed attribution methodology in terms of FPRs and FRNs provides a recognized measure that is common in the evaluation of anomaly detection schemes. In the context of a FRC attack, however, such an measure does not provide a sufficient amount of information for the cloud consumer to analyze the *risk* (i.e. risk = likelihood * impact) of a respective attack scenario, threshold value, and resulting FPR and FNRs [82, 113]. This section explores the results presented in Section 6.7 with the purpose of better understanding the *likelihood* of such attacks and enumerating the *impact* that they would have on a cloud

consumer.

6.8.1 Likelihood

The likelihood of an impactful FRC attack is largely dependent on two factors: the threat source and the vulnerability [82].

6.8.1.1 Vulnerability

The ease of discovering the utility model as an exploitable vulnerability should be easy for the attacker as the vulnerability has been published in a number of academic and non-academic forums and is literally hiding in plain sight. A potentially more difficult challenge for the threat source would be to determine whether or not a particular target is hosted in a CSP environment without the aide of an endorsement to a specific cloud vendor or a public acknowledgement. Once the threat source identifies the vulnerability, exploiting the utility model is simply a matter of making protocol adherent requests to the target website. As this document contains the seminal work in both FRC detection and attribution, current cloud consumers are unlikely to detect such an attack through technical measures. Outside inference of an attack based increases in request volume and/or the audit of the monthly bill.

6.8.1.2 Threat Source

As evidenced by the growing number, capacity, and sophistication of both botnets and DDoS attacks respectively [11], the worst-case threat source (i.e. botmaster) undoubtedly possesses the skill level to mount a FRC attack. As discussed in Section 3.4.2, the motive for performing such an attack ranges from ego and hacktivism to monetary gain, revenge, creating a competitive advantage and/or economic espionage [113]. Although a FRC attack could conceivably be carried out from any Internet-connected device, the worst-case scenario is the attack origination from a large bot. An objective this risk analysis seeks to address is the minimum botnet size, given the results presented in Section 6.7, that an threat source must be in control of in order to eclipse the threshold of nuisance activity and perform a successful FRC attack.

Determining the minimum size of a botnet needed to carry out a FRC attack is a function of the target's detection rate (i.e FNR) and cost threshold (i.e. J1). Tables 6.9, 6.10, and 6.11 display the resulting cost of a FRC attack for botnets consisting in size of 10,000, 100,000, and 500,000 respectively. The first column in these graphs represents the FNR from the attribution methodology in Section 6.6 and the row heading denotes the average number of request each individual bot client issues. For example, as reported in Table 6.9 for a 10,000 node botnet, if each client is to issue 20 requests and the FNR is 10.0% then the resultant monthly cost to the cloud consumer would be $0.92. As the number of requests and corresponding FNRs increase, so do the fraudulent costs.

Table 6.9: Monthly Costs of a FRC Attack ($) - 10,000 Client Botnet

	Average Requests per Attacker									
FNR	20	40	60	80	100	120	140	160	180	200
10.0	0.92	1.83	2.75	3.66	4.58	5.49	6.41	7.32	8.24	9.16
20.0	1.83	3.66	5.49	7.32	9.16	10.99	12.82	14.65	16.48	18.31
30.0	2.75	5.49	8.24	10.99	13.73	16.48	19.23	21.97	24.72	27.47
40.0	3.66	7.32	10.99	14.65	18.31	21.97	25.63	29.30	32.96	36.62
50.0	4.58	9.16	13.73	18.31	22.89	27.47	32.04	36.62	41.20	45.78
60.0	5.49	10.99	16.48	21.97	27.47	32.96	38.45	43.95	49.44	54.93
70.0	6.41	12.82	19.23	25.63	32.04	38.45	44.86	51.27	57.68	64.09
80.0	7.32	14.65	21.97	29.30	36.62	43.95	51.27	58.59	65.92	73.24
90.0	8.24	16.48	24.72	32.96	41.20	49.44	57.68	65.92	74.16	82.40
100.0	9.16	18.31	27.47	36.62	45.78	54.93	64.09	73.24	82.40	91.55

From the analysis of the 10,000 node botnet, it can be seen that a cloud instance that does not deploy any sort of defense against a FRC attack (i.e. FNR = 100%) that the resultant costs are likely to be insignificant to the cloud consumer - if the individual bot request rate is under 200 documents a month - and thus would be classified as nuisance activity. Such analysis is, however, relative to each cloud consumer and their budgeted monthly costs and expected data usage. For a FNR=30%, even at 200 requests per bot client over a given month, the resultant costs would be approximately $27.47. For a FRC attacker to be successful, even against a Trace-Driven attack, a 10,000 node botnet would not be sufficient unless clients requested a significant amount of web documents larger than what was analyzed (i.e. greater than 200

requests per client in a given month).

Increasing the size of the attack botnet subsequently increases the risk of a FRC attack. While a 100,000 node botnet is certainly more rare than a 10,000 node botnet, a FRC attacker is not restrict to performing the attack from a single botnet. Recent analysis has suggested botnet owners are aware that larger botnets draw attention from law enforcement agency and security professions and in turn attackers have reacted by creating smaller botnets of less than 50,000 nodes to remain hidden [11]. As a result, a FRC attack could be conducted from many smaller botnets and together the aggregate clients from these botnets could amass to much larger numbers.

Comparing the presented attack costs in Table 6.9 with that in Table 6.10, it can be seen that the financial impact of a FRC attack is linearly proportional to the resulting size of the botnet. Given the costs in Table 6.10 for a 100,000 node botnet requesting 200 documents per client, the FRC attack has the potential to transition from nuisance activity into the FRC attack region. Given these stated attack parameters, the resultant attack rate needed to sustain such an attack would be 7.71 requests per second, well within the performance capacity of a CSP server instance.

Table 6.10: Monthly Costs of a FRC Attack ($) - 100,000 Client Botnet

	Average Requests per Attacker									
FNR	20	40	60	80	100	120	140	160	180	200
10.0	9.16	18.31	27.47	36.62	45.78	54.93	64.09	73.24	82.40	91.55
20.0	18.31	36.62	54.93	73.24	91.55	109.86	128.17	146.48	164.79	183.11
30.0	27.47	54.93	82.40	109.86	137.33	164.79	192.26	219.73	247.19	274.66
40.0	36.62	73.24	109.86	146.48	183.11	219.73	256.35	292.97	329.59	366.21
50.0	45.78	91.55	137.33	183.11	228.88	274.66	320.43	366.21	411.99	457.76
60.0	54.93	109.86	164.79	219.73	274.66	329.59	384.52	439.45	494.38	549.32
70.0	64.09	128.17	192.26	256.35	320.43	384.52	448.61	512.70	576.78	640.87
80.0	73.24	146.48	219.73	292.97	366.21	439.45	512.70	585.94	659.18	732.42
90.0	82.40	164.79	247.19	329.59	411.99	494.38	576.78	659.18	741.58	823.97
100.0	91.55	183.11	274.66	366.21	457.76	549.32	640.87	732.42	823.97	915.53

Increasing the size of the botnet by a factor of five does elevates the potential cost impact for a cloud consumer but it does so at the loss of subtlety. While the FRC attack considered in this work was done so under the guise that the attacker attempted to be as surreptitious as possible in order to avoid detection, this certainly is not the only concern. The utility model

is also vulnerable to blatant DDoS attacks in which the loss of availability to the victim is magnified by the high data usage costs. To enumerate this potential, the peak DDoS rate reported in 2011 was that of 45Gbps which was launched from a 250,000 node botnet over the course of a week [25]. If such a DDoS attack were performed on a CSP instance, assuming the target website was continued to operate in the cloud, among other considerations, the resultant costs to the cloud consumer would be approximately $0.68/s, $40/min, $2430/h, $58,320/day, and $408,240/week.

Table 6.11: Monthly Costs of a FRC Attack ($) - 500,000 Client Botnet

	Average Requests per Attacker									
FNR	20	40	60	80	100	120	140	160	180	200
10.0	45.78	91.55	137.33	183.11	228.88	274.66	320.43	366.21	411.99	457.76
20.0	91.55	183.11	274.66	366.21	457.76	549.32	640.87	732.42	823.97	915.53
30.0	137.33	274.66	411.99	549.32	686.65	823.97	961.30	1098.63	1235.96	1373.29
40.0	183.11	366.21	549.32	732.42	915.53	1098.63	1281.74	1464.84	1647.95	1831.05
50.0	228.88	457.76	686.65	915.53	1144.41	1373.29	1602.17	1831.05	2059.94	2288.82
60.0	274.66	549.32	823.97	1098.63	1373.29	1647.95	1922.61	2197.27	2471.92	2746.58
70.0	320.43	640.87	961.30	1281.74	1602.17	1922.61	2243.04	2563.48	2883.91	3204.35
80.0	366.21	732.42	1098.63	1464.84	1831.05	2197.27	2563.48	2929.69	3295.90	3662.11
90.0	411.99	823.97	1235.96	1647.95	2059.94	2471.92	2883.91	3295.90	3707.89	4119.87
100.0	411.99	823.97	1235.96	1647.95	2059.94	2471.92	2883.91	3295.90	3707.89	4119.87

From the analysis of the threat source, given the implementation of presented attribution scheme and in the context of the two presented datasets, a FRC attacker is going to be required to possess control of a botnet near the size of 500,000 bots or larger to carry out a successful attack. While results will vary from site to site, perhaps the simplest measure to take to decrease the risk of a FRC attack is to limit the amount of requests a single client can enact in a time certain period of time regardless of the affect that the client has on the QoS of the web server or network.

As a result of the cloud utility vulnerability being openly published and conceivably easily discoverable, the likelihood of a FRC attack is largely dependent on the motive of the threat source and the size of the attacking botnet. As evidenced by the provided analysis, the potential impact of a FRC attack can be significant, adding to the overall risk of a FRC attack.

6.8.2 Impact

For the cloud consumer, the impact of a FRC attack can be broken down in to *technical impact factors* and *business impact factors*.

6.8.2.1 Technical Impact Factors

Following the exploitation of the cloud utility model, the predominant concern for the victim would not be the loss of confidentiality or integrity of data but instead the loss of the availability or financial viability of hosting public web content in the cloud. When coupled with the concern of vendor lock-in (Section 2.6.4), a FRC attack could potentially lead to not only significant monetary costs but also extensive service interruptions while the cloud consumer extracts and relocates their web-based business presence.

6.8.3 Business Impact Factors

While the predominant impact of a FRC attack has been described as a financial loss due to the exploitation of the utility model, the data usage costs are not the only toll a FRC attack inflicts on its victim. In addition to bandwidth costs, a cloud consumer must also consider auxiliary technical costs of attempting to mitigate a FRC attack as well as the indirect costs that such measures will have on legitimate users as a result of the inevitable FPs presented in this chapter. Weighing these factors, the cloud consumer must ultimately determine whether the business risk justifies investing in mitigation solutions.

The task of assessing the severity of the risk associated to a FRC attack must be done on an individual cloud consumer basis. The conclusions that can be draw from this FRC risk analysis is that there exist motivated threat sources highly capable of exploiting the utility pricing model. Given the both the direct and indirect costs of a FRC attack, the impact has the potential to be severe.

6.9 Discussion and Future Work

Presented in this paper is a reactive solution to the FRC attack based on the principles of anomaly detection. By assuming that each client is innocent until proven guilty, the major limitation to this approach is if an attacker can learn normal request patterns. As with any anomaly detection based scheme, when the attackers actions are no longer anomalous, the methodology is rendered ineffective. Although not directly explored in this work, an upper bound can be placed on the request volume as an additional detection safeguard. For example, if an attack client consumes more than 50 web documents and the client's usage footprint is not flagged as malicious, the attack client is then subjected to some form of challenge or hindrance. The inevitable downside to this approach is that normal clients requesting over 50 documents will also be burdened by the same mitigation technique.

Beyond pursuing attack attribution, a cloud consumer could take a proactive stance by limiting the available web content that is publicly accessible. This is commonly performed through user authentication, access control and serving graphical puzzles to limit readily available web content. However, these solutions are not without their faults as they are not immune to attack and they themselves frustrate legitimate user who my be unable or unwilling to engage with the control. These user frustration considerations are not new with the introduction of the FRC attack, however the cost benefit equation has shifted knowing that freely available content in the Cloud could cost the cloud consumer even more in terms of operational expenses.

Given this initial work on FRC attribution, there are several avenues of future work to explore. While optimized attack strategies were not considered as part of the threat model in this work, a future endeavor will be to consider the data usage footprint of each client. In this case, the attacker seeks to minimize the volume of requests by choosing the largest web documents hosted. Although such an attack would still likely be identifiable by presented attribution methodology, examining individuals' data usage provides an addition dimension for limiting the risk of a FRC attack. Similarly, other aspects of observed client behavior such as the use of client-side caching, or lack thereof, and the categorization and distributions of both primary and secondary requests will also be explored. In addition, diurnal, daily, weekly and

seasonal traffic patterns will also be considered to better characterize normal client behaviors.

In this work it is also assumed that the cloud consumer is able to prime the attribution methodology with a given period of training days. Future work will also explore non-supervised machine learning techniques when the cloud consumer is not afforded such a luxury.

6.10 Conclusion

The pay-as-you-go pricing model introduces a vulnerability into current CSP offerings. Web content hosted under a CSP's utility model enables an attacker to perform a FRC attack by simply making protocol compliant requests. Due to the ease of exploitation, the only factor preventing a FRC attack is the threat source's motivation. With limited prevention controls available, attribution methodologies are necessary for cloud consumers to defend their commitments to cloud computing. Based on the results from three progressively challenging attack scenarios, the proposed attribution methodology has achieved qualified success.

CHAPTER 7. PREVENTION AND MITIGATION

Chapter contains modified content from the following submitted journal paper:

Idziorek, J., Tannian, M. and Jacobson, D. Insecurity of Cloud Utility Models. *IEEE IT Professional*, © IEEE 2012.

The predominant focus of this work is on the detection and attribution of FRC attacks on public cloud utility models. Bookends between these to aspects of security are prevention and mitigation respectively. This chapter briefly discusses methodologies that cloud be used to quell a FRC attack before it comes to fruition and how a cloud consumer could respond once a FRC attack has been detected and individual malicious clients have been identified.

7.1 Prevention

A common way to prevent the exploitation of a vulnerability is to download and apply a patch for it. However, in the context of the utility pricing model, the bug is not a software defect but a common business model deployed by CSPs. Until this vulnerability is exploited on a frequent basis, the cloud business model is not likely to change. In lieu of a patch for this vulnerability there are several, albeit limited, prevention options.

7.1.1 User Authentication

The first option for the cloud consumer would be to limit the amount of readily available public-facing, and thus exploitable, content by implementing a user authentication control into a given website. By first requiring a user to authenticate to gain access to the majority of a site's content lessens the consumable footprint available to a malicious client and potentially

lends itself well to a more focused attribution solution. Furthermore, authentication assists in the identification and mitigation of malicious clients as login credentials can be suspended or revoked independent of an IP address. There are trade-offs, however - by enabling a user authentication mechanism for content that is intended to be publicly viewable and indexable on the WWW, such an approach may lead to clients unwilling to go through the necessary steps to establish login credentials or clients not being able to search for the desired content in the first place as authentication prevents search engines from indexing content. Lastly, although the user authentication would potentially limit the amount of exploitable content, the login page itself and other publicly available content is not impervious to a FRC attack. Although user authentication may be effective in some contexts, this control is only viable if it is consistent with the web applications objectives.

7.1.2 Graphical Puzzels

The second option is to challenge those that initiate requests at the onset of a connection with a CAPTCHA like puzzle. Often used in the prevention of DDoS attacks, CAPTCHAs challenge the requesting client with a visual puzzle under the assumption that humans can easily solve the puzzle while bot computers can not. Again, while effective in some circumstances, such graphical puzzles often complicate the process for clients to obtain the information that they are seeking, causing some clients to lose interest. Similar to the user authentication prevention option, CAPTCHAs are not immune to attacks and a motivated attacker could easily established multiple user accounts or manually defeat CAPATCHAs to bypass this technical control. In essence, the two presented authentication controls are simply raising the bar of effort required by the attacker to perform a sustained and successful FRC attack.

7.1.3 Application Design

The third option for the cloud consumer is to work with application and content developers on minimizing the resource footprint of the common or average web object. Limiting the impact of client requests increases the costs for the FRC attacker and the risk of detection and attribution. As was shown in Chapter 6, the more requests a individual client invokes, the

more anomalous they become. While such a solution certainly does not completely obviate a FRC attack, it does provide a more client-focused and transparent solution in comparison to the discussed authentication controls.

7.1.4 Web Hosting Environment

Lastly, if the perceived risk of a FRC attack coupled with other security concerns, such as vendor lock-in, is a serious concern to the cloud consumer, a potential prevention solution would be to simply host the web application in an environment not governed by a utility pricing model. Given the systems of analysis presented in Chapter 2, the cloud consumer must ultimately decide whether the benefits and cost savings of moving to the cloud are worth the risks as such an act is a voluntary business decision.

Unfortunately, without a utility model patch, the presented controls will not thwart a motivated attacker. Facing limited prevention capability, the next line of defense is the detection of FRC attacks as presented in Chapter 5.

7.2 Mitigation

Once a FRC attack has been detected and malicious clients have been identified, it is advantageous that the cloud consumer has a mitigation strategy or policy in place to deal with the suspected malicious actors. When formulating mitigation solutions, the cloud consumer must consider the potential for legitimate clients being errantly classified as malicious, which is an undesirable and largely unavoidable aspect of anomaly detection.

7.2.1 Attacker Identification

Malicious requests that compose a FRC attack are likely to originate from zombie or bot computers that unwittingly and surreptitiously participate in an attack unbeknownst to their owners. As a result of an anomaly-based attribution scheme, mitigation of a FRC attack could be accomplished by serving a suspected client a graphical puzzle to prove that the client is indeed a human. In this context, the graphical puzzle is being served to a potential attack

client *after* the defender has reason to believe the client is malicious. This technique has also been proposed as a mitigation solution for DDoS attacks [58].

Because it is in the best interest of attack clients to form fully-completed TCP connections in order to attack application-layer resources, attack clients are prevented from spoofing their IP address. While knowing the true IP address of an attack client aides in the process of attribution and mitigation, the defender has limited options to actually act on this information to resolve the problem at the source of the attack. Instead of seeking out individual bots, the defender is better off filtering or rate-limiting the offending IP address.

7.2.2 Filtering

While heavy-handed approaches like blacklisting first-time offenders may prove to do more harm than good given the ever-present FPR rates, filtering offending attackers is a realistic strategy to decrease the financial impact of attack clients. In other contexts, filtering options have been employed using a "three strikes and you're out" rule.

7.2.3 Rate Limiting

In contrast to filtering, less absolute mitigation strategies include assessing anomalous clients a back-off timeout in which requests from an IP address are not all serviced. Clients, whether malicious or not, could also be given a data usage quote. Like many popular news websites, after a quota has been reached the requesting client is either required to authenticate, given restricted access, or is denied service all together.

These reactive approaches discussed in this section are available today, but with limited detection and attribution solutions available to cloud consumers, the deployment and maintenance of such solutions will be challenging.

CHAPTER 8. CONCLUSION

8.1 Future Work

The impact this work will have on the larger research and cloud communities will be highly dependent on the future work and understanding of this vulnerability by other researchers and security practitioners. While the solutions presented in this work are novel and highly effective given the presented attack scenarios, one must be reminded that these are also the seminal works on this specific topic. If history is any indicator, the critique and reformulating of these ideas through several iterations or perhaps new approaches will be needed to address the presented vulnerability of the cloud utility model.

The body of ideas discussed in this work fundamentally change the way researchers and security practitioners address anomaly detection of web traffic. With the introduction of the utility model, much more subtle attack behaviors can have an impactful effect and thus the analysis of such subtle attacks has opened the door for new research ideas, methodologies and contributions. In addition to addressing the FRC attack, the detection and attribution methodologies and metrics presented are relevant beyond the state context and could also be applicable in other areas of study including DDoS detection and web usage mining.

Apart from the chapter specific areas of Future work addressed in Sections 4.8, 5.10, and 6.9, a key component that will enable future work in area of FRC detection and attribution will be for security researchers to have access to a diversity of web logs from live websites. Although historical analysis of web logs is beneficial, much more interesting and comprehensive analysis can be performed given the structure of a live website.

8.2 Contributions

The contributions of this dissertation are: (1) Conducted a broad security assessment of the public cloud computing model by applying the Parkerian Hexad as a system of analysis; (2) Provided detailed evidence that the utility pricing model is an unexplored and unaddressed vulnerability of the public cloud computing model; (3) Formalized a threat model and clearly defined the Fraudulent Resource Consumption (FRC) as an attack that exploits the utility model vulnerability; (4) Examined, differentiated, and applied the relevant bodies of work in related fields of study to the analysis of the FRC attack; (5) Developed a web traffic generation algorithm and accompanying characterization metrics to simulate and model the worst-case FRC attack scenario; (6) Formulated a FRC detection methodology by utilizing three qualities of aggregate web traffic to distinguish normal web traffic from that of a FRC attack; (7) Devised a FRC attribution solution by characterizing four aspects of individual client request behavior to identify normal clients from that of FRC attack clients.

8.3 Summary

As they are structured today, cloud utility models are vulnerable to exploitation. By allowing any client with access to the Internet to consume resources that are in turn metered and billed exposes the cloud consumer to a risk that is only mitigated by time, detection, attribution and accountability. Apart from the solutions presented in this dissertation, there have been no previously known defensive strategies. In addition to technical measures, awareness and understanding of this vulnerability are a key means of defense, and the works that composes this document has strived to achieve those goals. Unless utility models are restructured to remove the threat of a FRC attack, research in detection and attribution is necessary to ensure long-term sustainability of cloud consumers and remove one more impediment that could dissuade organizations from adopting public cloud computing.

To date, there have been no known public acknowledgements of a FRC attack occurring on the public cloud. The absence of such knowledge, however, does not confirm that such a vulnerability has not or will not be exploited. As an analog, back in the early 1990's,

Internet-facing firewalls were new and thought to be sufficient to secure a connected enterprise. However, reality was that attacks were occurring and intrusion detection systems soon pointed out these threats. Perhaps the utility model has been exploited and, as an security and research community, we are presently ill equipped to detect its presence or identify its culprits.

BIBLIOGRAPHY

[1] Amazon. Amazon S3 Availability Event. http://status.aws.amazon.com/s3-20080720.html, July 2008.

[2] Amazon. Amazon EC2 Service Level Agreement. http://aws.amazon.com/ec2-sla/, November 2010.

[3] Amazon. Case Studies. http://aws.amazon.com/solutions/case-studies/, September 2010.

[4] Amazon Web Services. Amazon EC2 Pricing. http://aws.amazon.com/ec2/pricing/, November 2011.

[5] Amazon Web Services. Amazon EC2 Instance Types. http://aws.amazon.com/ec2/instance-types/, February 2012.

[6] Apache. Apache HTTP Server Benchmarking Tool. http://httpd.apache.org/docs/2.0/programs/ab.html, January 2012.

[7] Martin F. Arlitt and Carey L. Williamson. Web Server Workload Characterization: The Search for Invariants. *SIGMETRICS Performance Evaluation Review*, 24:126–137, May 1996.

[8] Karyn Benson, Rafael Dowsley, and Hovav Shacham. Do You Know Where Your Cloud Files Are? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 73–82, New York, NY, USA, 2011. ACM.

[9] Sören Bleikertz, Thomas Groß, and Sebastian Mödersheim. Automated verification of

virtualized infrastructures. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 47–58, New York, NY, USA, 2011. ACM.

[10] Jose Borges and Mark Levene. Evaluating variable-length markov chain models for analysis of user web navigation sessions. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):441–452, April 2007.

[11] Mark Bowden. *Worm: The First Digital World War*. Grove/Atlantic, Inc., 2011.

[12] Tony Bradley. Google Outages Damage Cloud Credibility. http://www.pcworld.com/businesscenter/article/172614/google_outages_damage_cloud_credibility.html, September 2009.

[13] Mary Brandel. Cloud Computing: Dont Get Caught Without an Exit Strategy. http://www.computerworld.com/s/article/9128665/, March 2009.

[14] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In IEEE Communications Society, editor, *Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE INFOCOM '99.*, volume 1, pages 126–134, March 1999.

[15] Jon Brodkin. Cloud Computing Outages: Amazon Customers the Latest to Suffer Downtime. http://www.networkworld.com/community/node/48961, December 2009.

[16] Jon Brodkin. Microsoft issues service credits after cloud outage. http://www.networkworld.com/news/2010/091310-microsoft-cloud-outage.html, September 2010.

[17] Susanne Burklen, Pedro Jose Marron, Serena Fritsch, and Kurt Rothermel. User Centric Walk: An Integrated Approach for Modeling the Browsing Behavior of Users on the Web. In *Proceedings of the 38th annual Symposium on Simulation*, pages 149–159, 2005.

[18] Elie Bursztein, Steven Bethard, Celine Fabry, John C. Mitchell, and Dan Jurafsky. How Good are Humans at Solving CAPTCHAs? A Large Scale Evaluation. In *2010 IEEE Symposium on Security and Privacy*, pages 399–413, 2010.

[19] Jin Cao, William S. Cleveland, Yuan Gao, Kevin Jeffay, F. Donelson Smith, and Michele Weigle. Stochastic Models for Generating Synthetic HTTP Source Traffic. In *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2004*, volume 3, pages 1546–1557, March 2004.

[20] Xin Chen and Xiaodong Zhang. A Popularity-Based Prediction Model for Web Prefetching. *IEEE Computer*, 36(3):63–70, March 2003.

[21] Shuxing Cheng, Carl K. Chang, and Liang-Jie Zhang. Stochastic Modeling Study for Competitive Web Services Market. In *IEEE International Conference on Web Services*, pages 960–967, July 2007.

[22] Richard Chow, Philippe Golle, Markus Jakobsson, Elaine Shi, Jessica Staddon, Ryusuke Masuoka, and Jesus Molina. Controlling Data in the Cloud: Outsourcing Computation Without Outsourcing Control. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, pages 85–90, New York, NY, USA, 2009. ACM.

[23] ClarkNet-HTTP. http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html, 1995.

[24] Reuven Cohen. Cloud attack: Economic denial of sustainability (edos). http://groups.google.com/group/cloudforum/browse_thread/thread/8eeb522dd378edb7?pli=1, 2009.

[25] Lucian Constantin. Largest DDoS Attack so Far This Year Peaked at 45 Gbps, Says Company. http://www.networkworld.com/news/2011/112411-largest-ddos-attack-so-far-253462.html, November 2011.

[26] Lucian Constantin. Denial-of-service Attacks Are on the Rise, Anti-DDoS Vendors Report. http://www.pcworld.com/businesscenter/article/249438/denialofservice_attacks_are_on_the_rise_antiddos_vendors_report.html, February 2012.

[27] Mark E. Crovella and Azer Bestavros. Self-Similarity in World Wide Web traffic: Evi-

dence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5:835–846, December 1997.

[28] Dorothy Elizabeth Robling Denning. *Information Warfare and Security*. ACM Press Series. ACM Press, 1999.

[29] Stephen Dill, Ravi Kumar, Kevin S. Mccurley, Sridhar Rajagopalan, D. Sivakumar, and Andrew Tomkins. Self-Similarity in the Web. *ACM Transactions on Internet Technology*, 2:205–223, August 2002.

[30] Andrew Donoghue. Lightning Zaps Amazon Cloud. http://news.cnet.com/8301-1001_3-10263425-92.html, June 2009.

[31] George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza A. Basrai, and Peter M. Chen. ReVirt: Enabling Intrusion Analysis Through Virtual-Machine Logging and Replay. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, OSDI '02, pages 211–224, New York, NY, USA, 2002. ACM.

[32] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank Aggregation Methods for the Web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 613–622. ACM, 2001.

[33] Victor Echeverria, Lorie M. Liebrock, and Dongwan Shin. Permission Management System: Permission as a Service in Cloud Computing. In *IEEE 34th Annual Computer Software and Applications Conference Workshops (COMPSACW)*, pages 371–375, July 2010.

[34] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. Statistical Approaches to DDoS Attack Detection and Response. In *In Proceedings of the DARPA Information Survivability Conference and Exposition*, volume 1, pages 303–314, April 2003.

[35] Jun Feng, Yu Chen, Wei-Shinn Ku, and Pu Liu. Analysis of Integrity Vulnerabilities

and a Non-repudiation Protocol for Cloud Data Storage Platforms. In *39th International Conference on Parallel Processing Workshops (ICPPW)*, pages 251–258, September 2010.

[36] Mary Pat Flaherty. Bethesda Man Guilty in $4 Million Pay Phone Scams. http://www.washingtonpost.com/blogs/crime-scene/post/bethesda-man-guilty-in-4-million-pay-phone-scam/2011/09/21/gIQAfRrElK_blog.html, September 2011.

[37] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud Computing and Grid Computing 360-Degree Compared. In *Grid Computing Environments Workshop*, pages 1–10, November 2008.

[38] Tal Garfinkel and Mendel Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *In Proceedings of Network and Distributed Systems Security Symposium*, pages 191–206, 2003.

[39] Owen Garrett. Keeping Cloud Costs Grounded. http://www.forbes.com/2010/06/02/internet-software-zeus-technology-cloud-computing-10-garrett.html, June 2010.

[40] Dimitris Gavrilis, Ioannis Chatzis, and Evangelos Dermatas. Flash Crowd Detection Using Decoy Hyperlinks. In *2007 IEEE International Conference on Networking, Sensing and Control*, pages 466–470, April 2007.

[41] Roger Gellman. Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing. http://www.worldprivacyforum.org/pdf/WPF_Cloud_Privacy_Report.pdf, February 2009.

[42] Frank Gens. New IDC IT Cloud Services Survey: Top Benefits and Challenges. www.http://blogs.idc.com/ie/?p=730, December 2009.

[43] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[44] Dan Goodin. Webhost Hack Wipes Out Data for 100,000 Sites. http://www.theregister.co.uk/2009/06/08/webhost_attack/, June 2009.

[45] Google Apps. Google Apps Service Level Agreement. http://www.google.com/apps/intl/en/terms/sla.html, February 2012.

[46] Tim Greene. Unchecked Usage Can Kill Cost Benefits of Cloud Services. http://www.cio.com/article/682289/Unchecked_Usage_Can_Kill_Cost_Benefits_of_Cloud_Services?page=1&taxonomyId=3045, May 2011.

[47] Grant Gross. Obama Internet Kill Switch Plan Approved by US Senate Panel. http://news.techworld.com/security/3228198/obama-internet-kill-switch-planapproved-by-us-senate/, June 2010.

[48] Nils Gruschka and Meiko Jensen. Attack Surfaces: A Taxonomy for Attacks on Cloud Services. In *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pages 276–279, July 2010.

[49] Zhiyun Guo, Meina Song, and Junde Song. A Governance Model for Cloud Computing. In *2010 International Conference on Management and Service Science (MASS)*, pages 1–6, August 2010.

[50] U. Gurav and R. Shaikh. Virtualization: A Key Feature of Cloud Computing. In *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*, ICWET '10, pages 227–229, New York, NY, USA, 2010. ACM.

[51] Christopher Hoff. Cloud Computing Security: From DDoS (Distributed Denial Of Service) to EDoS (Economic Denial of Sustainability). http://www.rationalsurvivability.com/blog/?p=66, November 2008.

[52] Shi-Ming Huang, David C. Yen, Luen-Wei Yang, and Jing-Shiuan Hua. An Investigation of Zipf's Law for Fraud Detection. *Decision Support Systems*, 46:70–83, December 2008.

[53] Wassim Itani, Ayman Kayssi, and Ali Chehab. Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures. In *Eighth IEEE International*

Conference on Dependable, Autonomic and Secure Computing (DASC'09), pages 711–716, December 2009.

[54] Mario Jensen, Jorg Schwenk, Nils Gruschka, and Luigi Lo Iacono. On Technical Security Issues in Cloud Computing. In *IEEE International Conference on Cloud Computing (CLOUD'09)*, pages 109–116, September 2009.

[55] Kaustubh R. Joshi, Guy Bunker, Farnam Jahanian, Aad van Moorsel, and Joseph Weinman. Dependability in the Cloud: Challenges and Opportunities. In *International Conference on Dependable Systems Networks*, pages 103–104, July 2009.

[56] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *Proceedings of the 11th International Conference on World Wide Web*, pages 293–304, New York, NY, USA, 2002. ACM.

[57] Balachandra Reddy Kandukuri, Ramakrishna V. Paturi, and Atanu Rakshit. Cloud Security Issues. In *IEEE International Conference on Services Computing (SCC'09)*, pages 517–520, September 2009.

[58] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur Berger. Botz-4-Sale: Surviving Organized DDoS Attacks that Mimic Flash Crowds. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 287–300, 2005.

[59] Michael L. Katz and Harvey S. Rosen. *Microeconomics*. Irwin/McGraw-Hill Advanced Series in Economics. McGraw-Hill, 1998.

[60] Charlie Kaufman. What's Different About Security in a Public Cloud? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 27–28. ACM, October 2011.

[61] Lori M. Kaufman. Data Security in the World of Cloud Computing. *Security Privacy, IEEE*, 7(4):61–64, July 2009.

[62] Lori M. Kaufman. Can a Trusted Environment Provide Security? *IEEE Security Privacy*, 8(1):50–52, January 2010.

[63] Lori M. Kaufman. Can Public-Cloud Security Meet Its Unique Challenges? *IEEE Security Privacy*, 8(4):55–57, July 2010.

[64] Samuel T. King and Peter M. Chen. SubVirt: Implementing Malware with Virtual Machines. In *2006 IEEE Symposium on Security and Privacy*, pages 14–27, May 2006.

[65] Isao Kotera, Ryusuke Egawa, Hiroyuki Takizawa, and Hiroaki Kobayashi. Modeling of cache access behavior based on zipf's law. In *Proceedings of the 9th workshop on MEmory performance: DEaling with Applications, systems and architecture*, MEDEA '08, pages 9–15, 2008.

[66] Michael Kretzschmar, Mario Golling, and Sebastian Hanigk. Security Management Areas in the Inter-cloud. In *2011 IEEE International Conference on Cloud Computing (CLOUD)*, pages 762–763, July 2011.

[67] Lukas Kroc, Stephan Eidenbenz, and James P. Smith. SessionSim: Activity-Based Session Generation for Network Simulation. In M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, editors, *Proceedings of the 2009 Winter Simulation Conference*, pages 3169–3180, Piscataway, New Jersey, December 2009. Institute of Electrical and Electronics Engineers, Inc.

[68] Christopher Kruegel and Giovanni Vigna. Anomaly Detection of Web-based Attacks. In *Proceedings of the 10th ACM conference on Computer and Communications Security*, CCS '03, pages 251–261, 2003.

[69] Nir Kshetri. The Economics of Click Fraud. *IEEE Security Privacy*, 8(3):45–53, May 2010.

[70] Quyen Le, Marat Zhanikeev, and Yoshiaki Tanaka. Methods of Distinguishing Flash Crowds from Spoofed DoS Attacks. In *3rd EuroNGI Conference on Next Generation Internet Networks*, pages 167–173, May 2007.

[71] Zhao Li and Jeff Tian. Testing the Suitability of Markov Chains as Web Usage Models. In *27th Annual International Computer Software and Applications Conference, COMPSAC 2003*, pages 356–361. IEEE Computer Society, November 2003.

[72] Zhen Li, Qi Liao, and Aaron Striegel. Botnet Economics: Uncertainty Matters. In *Managing Information Risk and the Economics of Security*, pages 245–267. Springer US, 2009.

[73] Huan Liu. A New Form of DOS Attack in a Cloud and its Avoidance Mechanism. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, CCSW '10, pages 65–76. ACM, 2010.

[74] Song Luo and Gerald A. Marin. Realistic Internet Traffic Simulation Through Mixture Modeling and a Case Study. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, editors, *Proceedings of the 2005 Winter Simulation Conference*, page 9, Piscataway, New Jersey, December 2005. Institute of Electrical and Electronics Engineers, Inc.

[75] Bruce A. Mah. An Empirical Model of HTTP Network Traffic. In *Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '97.*, volume 2, pages 592–600, April 1997.

[76] Umesh Maheshwari, Radek Vingralek, and William Shapiro. How to Build a Trusted Database System on Untrusted Storage. In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*, OSDI'00, page 10, Berkeley, CA, USA, 2000. USENIX Association.

[77] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing (Draft). http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf, February 2012.

[78] Cade Metz. DDoS Attack Rains Down on Amazon Cloud. http://www.theregister.co.uk/2009/10/05/amazon_bitbucket_outage/, October 2009.

[79] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can Homomorphic Encryption be Practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM.

[80] Kara Nance, Matt Bishop, and Brian Hay. Virtual Machine Introspection: Observation or Interference? *IEEE Security Privacy*, 6(5):32–37, September 2008.

[81] Kazuyuki Narisawa, Daisuke Ikeda, Yasuhiro Yamada, and Masayuki Takeda. Detecting Blog Spams Using the Vocabulary Size of All Substrings in Their Copies. In *In Proceedings of Workshop on Weblogging Ecosystem*, 2006.

[82] NASA-HTTP. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology, February 2012.

[83] NASA-HTTP. http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html, February 2012.

[84] Network World Staff. From Sidekick to Gmail: A Short History of Cloud Computing Outages. http://www.networkworld.com/news/2009/101209-sidekick-cloud-computingoutages-short-history.html, October 2009.

[85] Bhawna Nigam and Suresh Jain. Generating a New Model for Predicting the Next Accessed Web Page in Web Usage Mining. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology (ICETET)*, pages 485–490, November 2010.

[86] NIST. Tolerance Intervals for a Normal Distribution. http://www.itl.nist.gov/div898/handbook/prc/section2/prc263.htm, February 2012.

[87] G. Oikonomou and J. Mirkovic. Modeling Human Behavior for Defense Against Flash-Crowd Attacks. In *IEEE International Conference on Communications*, pages 1–6, 2009.

[88] Lewis Page. Join in the Wikileaks DDoS war from your iPhone or iPad. http://www.theregister.co.uk/2010/12/10/loic_for_iphone/, December 2010.

[89] Donn B. Parker. *Fighting Computer Crime*. Scribner, 1983.

[90] Gunnar Peterson. Don't Trust. And Verify: A Security Architecture Stack for the Cloud. *IEEE Security Privacy*, 8(5):83–86, September 2010.

[91] Guest Posts. When the Cloud Bursts Someone Gets Wet. http://www.cloudave.com/2544/when-the-cloud-bursts-someone-gets-wet/, January 2009.

[92] M. Požandenel, V. Mahnič and, and M. Kukar. Separation of Interleaved Web Sessions with Heuristic Search. In *2010 IEEE 10th International Conference on Data Mining (ICDM)*, pages 411–420, December 2010.

[93] Michael Price. The Paradox of Security in Virtual Environments. *IEEE Computer*, 41(11):22–28, November 2008.

[94] Rackspace. Bandwidth Pricing. http://www.rackspace.com/cloud/cloud_hosting_products/servers/pricing/, February 2012.

[95] Sreeram Ramachandran. Web Metrics: Size and Number of Resources. http://code.google.com/speed/articles/web-metrics.html, May 2010.

[96] S. Ramgovind, M.M. Eloff, and E. Smith. The Management of Security in Cloud Computing. In *Information Security for South Africa (ISSA)*, pages 1–7, August 2010.

[97] Supranamaya Ranjan, Ram Swaminathan, Mustafa Uysal, Antonio Nucci, and Edward Knightly. DDoS-shield: DDoS-resilient scheduling to counter application layer attacks. *IEEE/ACM Transactions on Networking*, 17:26–39, February 2009.

[98] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, You, Get Off of my Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 199–212, New York, NY, USA, 2009. ACM.

[99] John C. Roberts, II and Wasim Al-Hamdani. Who Can You Trust in the Cloud?: A Review of Security Issues Within Cloud Computing. In *Proceedings of the 2011 Information*

Security Curriculum Development Conference, InfoSecCD '11, pages 15–19, New York, NY, USA, 2011. ACM.

[100] Francisco Rocha, Salvador Abreu, and Miguel Correia. The Final Frontier: Confidentiality and Privacy in the Cloud. *IEEE Computer*, 44(9):44–50, September 2011.

[101] Farzad Sabahi. Cloud Computing Security Threats and Responses. In *IEEE 3rd International Conference on Communication Software and Networks*, pages 245–249, May 2011.

[102] Nuno Santos, Krishna P. Gummadi, and Rodrigo Rodrigues. Towards Trusted Cloud Computing. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[103] Karen Scarfone and Peter Mell. NIST Special Publication 800-94: Guide to Intrusion Detection and Prevention Systems (IDPS). http://www.csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf, February 2007.

[104] Joshua Schiffman, Thomas Moyer, Hayawardh Vijayakumar, Trent Jaeger, and Patrick McDaniel. Seeding Clouds with Trust Anchors. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, CCSW '10, pages 43–46, New York, NY, USA, October 2010. ACM.

[105] D. Sculley. Rank Aggregation for Similar Items. In *Proceedings of the Seventh SIAM International Conference on Data Mining*. SIAM, 2007.

[106] Shubhashis Sengupta, Vikrant Kaulgud, and Vibhu S. Sharma. Cloud Computing Security-Trends and Research Directions. In *2011 IEEE World Congress on Services*, pages 524–531, July 2011.

[107] Stephen Shankland. Google Docs Suffers Privacy Glitch. http://news.cnet.com/google-docs-suffers-privacy-glitch, March 2009.

[108] Zhidong Shen, Li Li, Fei Yan, and Xiaoping Wu. Cloud Computing System Based on Trusted Computing Platform. In *International Conference on Intelligent Computation Technology and Automation (ICICTA)*, volume 1, pages 942–945, May 2010.

[109] Zhidong Shen and Qiang Tong. The Security of Cloud Computing System Enabled by Trusted Computing Technology. In *2nd International Conference on Signal Processing Systems (ICSPS)*, volume 2, pages 11–15, July 2010.

[110] Alexander Shraer, Christian Cachin, Asaf Cidon, Idit Keidar, Yan Michalevsky, and Dani Shaket. Venus: Verification for Untrusted Cloud Storage. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, CCSW '10, pages 19–30, New York, NY, USA, 2010. ACM.

[111] MG Siegler. The Web Collapses Under The Weight Of Michael Jackson's Death. http://techcrunch.com/2009/06/25/the-web-collapses-under-the-weight-of-michael-jacksons-death/, June 2009.

[112] Juraj Somorovsky, Mario Heiderich, Meiko Jensen, Jörg Schwenk, Nils Gruschka, and Luigi Lo Iacono. All Your Clouds are Belong to Us: Security Analysis of Cloud Management Interfaces. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 3–14, New York, NY, USA, October 2011. ACM.

[113] Gary Stoneburner, Alice Goguen, and Alexis Feringa. NIST Special Publication 800-30: Risk Management Guide for Information Technology Systems. csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf, 2002.

[114] Phuoc Tran-Gia, Dirk Staehle, and Kenji Leibnitz. Source Traffic Modeling of Wireless Applications. *International Journal of Electronics and Communications*, 55(1):27–36, 2001.

[115] Bob Violino. Preparing for the Real Costs of Cloud Computing. http://www.computerworld.com/s/article/9218984/Preparing_for_the_real_costs_of_cloud_computing, August 2011.

[116] Luis von Ahn, Manuel Blum, and John Langford. Telling Humans and Computers Apart Automatically. *Communications of the ACM*, 47:56–60, February 2004.

[117] Phil Wainewright. Cogheads Demise Highlights PaaS Lock-out Risk. http://www.zdnet.com/blog/saas/cogheads-demise-highlightspaas-lock-out-risk/668, February 2009.

[118] Jin Wang, Xiaolong Yang, and Keping Long. A New Relative Entropy Based App-DDoS Detection Method. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 966–968, June 2010.

[119] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(5):847–859, May 2011.

[120] Sheng Wen, Weijia Jia, Wei Zhou, Wanlei Zhou, and Chuan Xu. CALD: Surviving Various Application-Layer DDoS Attacks That Mimic Flash Crowd. In *4th International Conference on Network and System Security (NSS)*, pages 247–254, 2010.

[121] Windows Azure. Service Level Agreements. http://www.microsoft.com/windowsazure/sla/, November 2010.

[122] Windows Azure. Pricing Overview. http://www.microsoft.com/windowsazure/pricing/, February 2012.

[123] Yi Xie and Shun-Zheng Yu. A Novel Model for Detecting Application Layer DDoS Attacks. In *First International Multi-Symposiums on Computer and Computational Sciences.*, volume 2, pages 56 –63, June 2006.

[124] Yi Xie and Shun-Zheng Yu. Monitoring the Application-Layer DDoS Attacks for Popular Websites. *IEEE/ACM Transactions on Networking*, 17(1):15–25, February 2009.

[125] Toshihiko Yamakami. A Zipf-Like Distribution of Popularity and Hits in the Mobile Web Pages with Short Life Time. In *Proceedings of the Seventh International Conference on*

Parallel and Distributed Computing, Applications and Technologies, PDCAT '06, pages 240–243, 2006.

[126] Jeff Yan and Ahmad Salah El Ahmad. A Low-Cost Attack on a Microsoft CAPTCHA. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 543–554, New York, NY, USA, 2008.

[127] Jinhui Yao, Shiping Chen, Surya Nepal, David Levy, and John Zic. TrustStore: Making Amazon S3 Trustworthy with Services Composition. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, pages 600–605, May 2010.

[128] K.H. Yeung and C.W. Szeto. On the Modeling of WWW Request Arrivals. In *1999 International Workshops on Parallel Processing*, pages 248–253, 1999.

[129] Jonathan Zdziarski. mod_evasive. http://www.zdziarski.com/blog/?page_id=442, 2011.

[130] Jie Zhang and Ali A. Ghorbani. The Reconstruction of User Sessions From a Server Log using Improved Time-Oriented Heuristics. In Ali A. Ghorbani, editor, *Second Annual Conference on Communication Networks and Services Research, 2004*, pages 315–322, May 2004.

[131] Bin B. Zhu, Jeff Yan, Qiujie Li, Chao Yang, Jia Liu, Ning Xu, Meng Yi, and Kaiwei Cai. Attacks and Design of Image Recognition CAPTCHAs. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 187–200, New York, NY, USA, 2010. ACM.

[132] George K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, 1949.