Aalto University
School of Science
Degree Programme in Security and Mobile Computing

Debopam Bhattacherjee

Stepping Stone Detection for Tracing Attack Sources in Software-Defined Networks

Master's Thesis
Espoo, June 30, 2016

Supervisors: Professor Tuomas Aura, Aalto University
Professor Markus Hidell, KTH Royal Institute of Technology

Advisor: Professor Andrei Gurtov

Aalto University
School of Science
Degree Programme in Security and Mobile Computing

ABSTRACT OF
MASTER'S THESIS

Author:	Debopam Bhattacherjee		
Title:	Stepping Stone Detection for Tracing Attack Sources in Software-Defined Networks		
Date:	June 30, 2016	**Pages:**	68
Major:	Security and Mobile Computing	**Code:**	T-110
Supervisors:	Professor Tuomas Aura Professor Markus Hidell		
Advisor:	Professor Andrei Gurtov		

Stepping stones are compromised hosts in a network which can be used by hackers and other malicious attackers to hide the origin of connections. Attackers hop from one compromised host to another to form a chain of stepping stones before launching attack on the actual victim host. Various timing and content based detection techniques have been proposed in the literature to trace back through a chain of stepping stones in order to identify the attacker. This has naturally led to evasive strategies such as shaping the traffic differently at each hop. The evasive techniques can also be detected.

Our study aims to adapt some of the existing stepping stone detection and anti-evasion techniques to software-defined networks which use network function virtualization. We have implemented the stepping-stone detection techniques in a simulated environment and use sFlow for the traffic monitoring at the switches. We evaluate the detection algorithms on different network topologies and analyze the results to gain insight on the effectiveness of the detection mechanisms. The selected detection techniques work well on relatively high packet sampling rates. However, new solutions will be needed for large SDN networks where the packet sampling rate needs to be lower.

Keywords:	Stepping stone attack, Software-defined networking, Network function virtualization
Language:	English

Aalto-universitetet
Högskolan för teknikvetenskaper
Examensprogram för Säkerhet samt Mobil Kommunikation

SAMMANDRAG AV
DIPLOMARBETET

Utfört av: Debopam Bhattacherjee

Arbetets namn:
Stepping Stone Detection för att spåra Attack Källor i Programvarustyrd Nätverk

Datum:	Den 30 Juni 2016	**Sidantal:**	68
Huvudämne:	Säkerhet samt Mobil Kommunikation	**Kod:**	T-110
Övervakare:	Professor Tuomas Aura Professor Markus Hidell		
Handledare:	Professor Andrei Gurtov		

Språngbrädor äventyras värdar i ett nätverk som kan användas av hackare och andra skadliga angripare att dölja ursprunget av anslutningar. Angripare hopp från en kompromitterad värd till en annan för att bilda en kedja av att kliva stenar innan lansera attack på själva offret värd. Olika timing och innehållsbaserad detekteringsteknik har föreslagits i litteraturen att spåra tillbaka genom en kedja av språngbräda för att identifiera angriparen. Detta har naturligtvis lett till undvikande strategier som forma trafiken annorlunda vid varje hopp. De undan tekniker kan också detekteras.

Vår studie syftar till att anpassa vissa av de befintliga inledande upptäckt och mot skatteflykt tekniker för mjukvarudefinierade nät som använder nätverksfunktionen virtualisering. Vi har genomfört de språngbräda detektionstekniker i en simulerad miljö och använda sFlow för trafikövervakning på växlarna. Vi utvärderar detekteringsalgoritmer på olika nätverkstopologier och analysera resultaten för att få insikt om hur effektiva mekanismer upptäckt. De valda detektionstekniker fungerar bra på relativt höga paketsamplingsfrekvenser. Dock kommer nya lösningar att behövas för stora SDN nätverk där paketsamplingshastigheten behöver vara lägre.

Nyckelord: Språngbräda attack, Mjukvarudefinierad nätverk, Nätverksfunktion virtualisering

Språk: Engelska

Acknowledgements

This work was supported by TEKES as part of the Cyber Trust program of DIGILE (the Finnish Strategic Center for Science, Technology and Innovation in the field of ICT and digital business).

I would like to thank my supervisor Prof. Tuomas Aura for providing his invaluable support and guidance. I am also equally thankful to Prof. Andrei Gurtov for providing useful advice and feedback on my work.

I am also grateful to my co-supervisor at KTH Royal Institute of Technology, Prof. Markus Hidell, for providing remote support.

Lastly, I would like to take this opportunity to thank my parents for their continuous support and guidance and my wife, Taniya, for being beside me through thick and thin.

Espoo, June 30, 2016

Debopam Bhattacherjee

Abbreviations and Acronyms

5G	5th Generation (mobile network)
APT	Advanced Persistent Threat
BaaS	Botnets-as-a-Service
C&C	Command-and-Control
DDOS	Distributed Denial of Service
DNS	Domain Name System
DVR	Digital Video Recorder
ForCES	Forwarding and Control Element Separation
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPD	Inter-packet delay
IPFIX	Internet Protocol Flow Information Export
IPS	Intrusion Prevention System
IRC	Internet Relay Chat
ISP	Internet Service Provider
LTE	Long-Term Evolution
LTE-A	Long-Term Evolution - Advanced
MIB	Management Information Base
NBI	North-Bound Interface
NetFlow	Network Flow
NFV	Network Function Virtualization
NOS	Network Operating System
NSC	Network Service Chaining
P2P	Peer-to-peer
RTT	Round-trip time
SBI	South-Bound Interface
SDN	Software-Defined Networking
sFlow	Sampled Flow

SSH	Secure Shell
TCP	Transmission Control Protocol
UMTS	Universal Mobile Telecommunications System
VLAN	Virtual Local Area Network
VoIP	Voice over IP
VPN	Virtual Private Network

Contents

Chapter 1

Introduction

Stepping stones [38] are compromised hosts in a network which can be used by attackers to evade detection. The attackers hop from one host to another before attacking the victim host in order to hide their identities. Efficient stepping stone detection techniques in the literature are able to identify the intermediate stepping stones and trace connections back to the host from which the attack originates. There are various stepping stone detection techniques [11, 20, 38, 55] which are able to detect stepping stones with high accuracy. Newer attack strategies include using botnets for launching distributed denial-of-service (DDOS) attacks or spamming. These strategies necessitate detection techniques which no more rely on traffic patterns generated due to human interaction. Various botnet detection techniques [29] such as deep packet inspection and scanning exist. Also the advent of anonymity networks like Tor [41] necessitates strong deanonymization techniques in case the attacker uses these networks for attacks.

Stepping stone detection techniques have not yet been evaluated or studied in the context of *software-defined networking* (SDN) [24] or *network function virtualization* (NFV) [2] environments. SDN is an easily programmable network architecture which separates the control plane from the data plane. A logically centralized controller uses the network-wide view provided by the network operating system (NOS) to effectively configure and control the data plane, which consists of forwarding elements (switches). NFV, on the other hand, aims to remove dependency on middle-boxes in networks by virtualizing network functions, which can be run on virtual appliances. Naturally, the stepping stone detection mechanisms must be adapted to these new environments. The significance lies in the fact that these technologies will be heavily used in the upcoming 5th generation (5G) mobile network architecture, and effective detection of attacks will make the network robust and trustworthy.

1.1 Research Problem

Our goal is *to analyze challenges of stepping stone detection in SDN and NFV environments, especially within upcoming 5G network architecture, as well as conduct practical experiments demonstrating detection.* We aim to achieve the following:

1. Analyze existing stepping stone detection techniques and their applicability to SDN and NFV based network architectures.

2. Propose and build an efficient SDN and NFV based architecture that supports the stepping stone detection mechanisms.

3. Evaluate the proposed architecture on various network topologies.

1.2 Research Methods

We first theoretically analyze the stepping stone attacks and their detection techniques. We then propose an architecture to support the detection techniques in SDN and NFV environments. In the experimental part, we implement the detection techniques and evaluate their effectiveness on various network topologies.

1.3 Impact and Sustainable Development

SDN and NFV are new avenues in computer networks. For wide-scale adoption of these techniques, it is necessary to make them robust and secure. Network monitoring plays a critical role in making any network robust. Identifying correlated connections is a part of network monitoring and helps in identifying stepping stones and tracing back to the attacker if there is any attack on the network devices. We believe that our study regarding detection of stepping stones in SDN and NFV environments is an essential component of network monitoring in SDN and NFV, which is essential for the success of these techniques in the word of networks. There are ethical impacts of our study on the society as we propose an architecture to detect criminal attackers. Also it is our responsibility as engineers to analyze and fix vulnerabilities of new technologies before they are deployed.

SDN and NFV enable low-cost installation and maintenance of networks. Due to the separation of the control plane and the data plane, the network devices have to perform less computation leading to energy savings.

Moreover, costly vendor-locked devices are no longer necessary and can be replaced by low-cost Linux boxes. Hence, wide adoption of SDN and NFV will lead to sustainable development of the technology and business. Upcoming 5G telecommunication networks will rely heavily on SDN and NFV. These networks will provide connectivity to millions of people who are not yet connected to mobile networks, thus reducing the communication latency for them.

1.4 Structure of the Thesis

The rest of the thesis is structured as follows: Chapter 2 introduces the concepts of stepping stone attacks, SDN, NFV and the significance of SDN and NFV in 5G networks. Chapter 3 discusses stepping stone attacks in further detail along with the existing detection techniques. Chapter 4 explains our approach to adapting the existing stepping stone detection techniques to SDN and NFV environments, as well as the experimental implementation. Chapter 5 presents the experimental results along with their interpretation while chapter 6 discusses the significance of the results along with the limitations of the current study and future directions. Chapter 7 concludes the thesis.

Chapter 2

Background

In this chapter, we give an overview of stepping stone attacks, SDN, NFV and the role of SDN and NFV in upcoming 5G networks.

2.1 Stepping Stone Attacks

In order to remain anonymous and evade detection, attackers establish long chains of connections from one compromised host to another and finally attack the victim host as shown in Figure 2.1. These intermediate compromised hosts are called *stepping stones*, and the family of attacks is known as stepping stone attacks. In order to identify the source of the attack, one needs to correlate the stepping stones in the chain and the connections between them. There are various ways in which stepping stones can be detected. Early detection techniques were content based [38]. The newer ones rely on timing-based detection techniques [1, 45, 54, 55]. This is because advanced attackers encrypt traffic at each intermediate node, which make detecting stepping stones based on correlated content impossible. Zhang et al. [55] propose a timing-based detection technique which relies on the interactive pattern of human typing, which generates traffic with periods when data flows and periods when there is no data. The former is called an ON period while the latter is called an OFF period. Connections are identified as correlated if their ON/OFF periods are highly correlated based on timing.

Jitter and Chaff Attackers can evade the timing-based stepping stone detection strategies by deliberately introducing random *jitter* and *chaff* in the generated traffic at some or all of the intermediate hosts. Random delay or jitter, introduced at an intermediate host, results in the distortion of ON/OFF period timings. Chaff packets or random padding added between

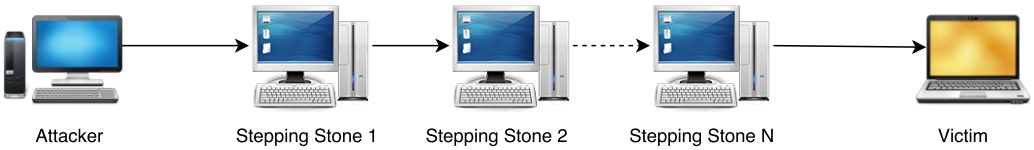

Figure 2.1: A typical stepping-stone attack.

two adjacent stepping stones result in increased number of ON periods or ON periods with longer duration. Both these techniques hinder the detection of stepping stones. Various anomaly detection techniques [11] have been proposed which detect jitter and chaff in interactive traffic. These can be used to augment the timing based stepping stone detection techniques.

2.2 Software-defined Networking

SDN is an emerging network architecture [31] where the data plane and the control plane are separated to make the network easily and dynamically configurable and programmable. As shown in Figure 2.2, the network architecture has 3 tiers: the application tier, the control tier and the infrastructure tier. The control tier consists of the logically centralized controller, which provides a network-wide view of the forwarding elements and their states to the application tier via north-bound interfaces (NBI). Distributed routing protocols are replaced in SDN by algorithms that make use of the global view of the network. The centralized control plane is the single point of configuration for the network administrators. The controller in turn manages the forwarding elements. Hence, the traffic can be dynamically shaped by the administrators without configuring the individual forwarding elements. NOX[1], POX[2], Floodlight[3] and OpenDaylight[4] are some widely used open-source SDN controllers.

The wide range of applications residing in the application tier, including load balancers, monitoring applications and intrusion detection systems use the network wide view provided by the control tier to monitor and control the data plane. These applications can use the NBI to specify network-level requirements to the controller. The controller translates the requirements into instructions for the forwarding elements.

[1]http://www.noxrepo.org/
[2]http://www.noxrepo.org/pox/about-pox/
[3]http://www.projectfloodlight.org/floodlight/
[4]https://www.opendaylight.org/

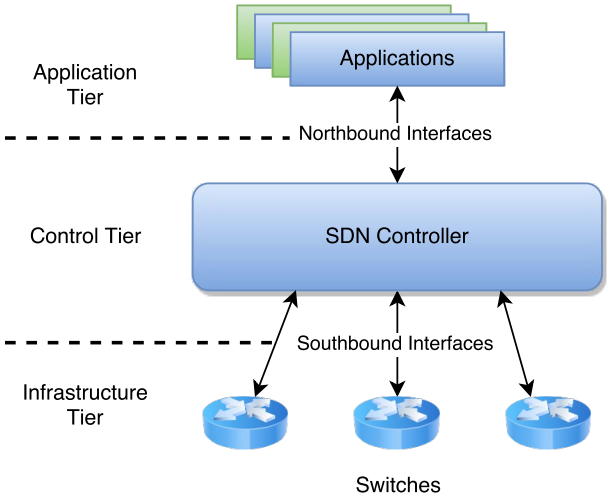

Figure 2.2: Overview of SDN architecture.

The infrastructure tier (data plane) consists of forwarding elements which typically forward packets based on layer-2 and layer-3 headers and are known as switches. The controller communicates with the switches using southbound interfaces (SBI). The SBI is used by the controller to send instructions to the switches and by the switches to consult the controller when they are not able to make the forwarding decision based on the previous instructions. Some of the well-known SBIs are OpenFlow [30], Forwarding and Control Element Separation (ForCES) [13] and SoftRouter [25].

2.2.1 OpenFlow

OpenFlow is currently the most widely used SBI. OpenFlow controllers communicate with OpenFlow compliant switches in the data plane through a control channel specified by the OpenFlow standard [3]. The controller installs flow entries to the flow tables of the switches. Each flow table entry contains match header fields, counters and actions (forward, drop, modify fields, etc.). The switches match incoming packets with the flow table entries, increment the counters and take the corresponding actions. The controller can control the routing behaviour of the switches by inserting, updating or deleting flow table entries.

2.2.2 sFlow

sFlow [34] is a traffic monitoring technique in networks. Low cost sFlow agents are installed in the switches which sample packets and forward sampled data to a data collector for analysis. sFlow defines the sampling techniques used in the sFlow agents, the sFlow management information base (MIB) used by the sFlow collector (analyzer) to control the sFlow agents and the format of the data forwarded by the sFlow agents to the collector.

OpenFlow and sFlow are complementary technologies. OpenFlow is an SBI for the SDN environments which configures the switches by translating user requirements into instructions and installing flow entries into the flow tables of the switches. sFlow, on the other hand, provides an API for network monitoring and opens up the possibility of performance aware network management and provisioning.

Cisco's NetFlow [8] and IPFIX (IETF's alternative to NetFlow) [9] serve a similar purpose to sFlow by forwarding flow records to an analyzer. A flow in NetFlow and IPFIX context refers to the set of packets with similar attributes. The switches sample packets, decode the headers to retrieve values of header fields like the source and destination IP addresses and source and destination ports, hash the decoded values to identify the flow in the flow cache, and update the flow with new values. On termination of the flows, the flow records are flushed from the cache and forwarded to the analyzer.

sFlow[5], in contrast to NetFlow and IPFIX, samples packets and forwards the sampled header information to the collector. The collector is responsible for decoding and analyzing the data. sFlow also provides a polling mechanism which periodically sends the values of the interface counters to the collector. This simplified sampling and forwarding mechanism reduces the performance overhead in the switches. Thus, sFlow is lightweight and does not consume resources by maintaining a flow cache in the switches. Its design also emphasizes scalability.

2.3 Network Function Virtualization

Network functions include switching, tunnelling, monitoring, service assurance, signalling and security functions. These network functions are implemented in proprietary hardware appliances which consume space and power in the network. Moreover, these hardware boxes require special skills to be administered and may result in vendor lock-in. Network function virtualization or NFV [5] aims to virtualize these network functions by leveraging

[5]http://blog.sflow.com/2012/05/software-defined-networking.html

virtualization techniques and commodity hardware. NFV facilitates the entry of software players in the networking market. Virtual network functions are built with software running on commodity hardware. These network functions can be instantiated in any part of the network without installing specialized hardware. NFV makes scaling up, scaling down and evolution of the network functions more flexible.

NFV and SDN are complimentary but independent concepts. NFV aims to provide virtualized network functions while SDN aims to separate the control and data planes in the network. Both these technologies aim to enhance network performance, simplify maintenance and dynamically control and provision network resources. Both NFV and SDN aim to use commodity hardware and switches to lower the overall cost of networking.

2.4 SDN and NFV in 5G

Fifth generation mobile network or 5G [27] is the next generation of telecommunication networks that is expected to provide extremely high bandwidth, low latency and highly robust connectivity to human users as well as the Internet of Things. Interconnections between the cellular networks and other wireless access infrastructures, forming heterogeneous networks (HetNets), will characterize 5G networks. Integrating satellite communication and supplying data from distributed sources to cloud-based big-data applications are some of the challenges 5G aims to solve. Robustness and resilience are necessary in order to support this varying range of services efficiently.

Adoption of SDN in mobile networks [19] helps to isolate the data plane from the control plane and eases the development of applications that provide network-level services at the application tier via NBI. The SDN controller performs network management using its global view of the network. This enables dynamic on-demand allocation of resources and network virtualization. NFV aims to replace dedicated hardware devices by software-based network function implementations deployed in virtualized infrastructure. The network providers can easily roll out new services on these hyper-flexible and programmable networks.

SDN and NFV may be used to provide network service chaining [19], which aims to provide chains of services in the network processing path. As SDN pulls out the management functions from network devices and places them in a software-based controller and NFV pulls out the network functions from hardware devices and builds them into software running on commodity servers, no additional hardware is required to provide network service chaining in SDN and NFV environments. Instead, the chaining can be im-

plemented and configured in software.

Chapter 3

Stepping Stone Attacks

In this chapter we describe stepping stone attacks in detail, discuss the relevance of the stepping stone attacks in today's networks and the various techniques for detecting stepping stones. Finally, we identify the significance of the attacks in SDN and NFV environments.

3.1 Significance in Today's Internet

Stepping stone attacks have existed since the early days of the Internet. Attackers try to hide their identity behind a chain of intermediate nodes compromised earlier while launching attacks on further victims. Also, an external intruder might compromise one host in an administered network by exploiting some vulnerabilities and use the host as a launch pad to gain useful insight about the network and hosts lying within it. Intrusion detection systems (IDS) and forensic analysis try to identify the node from which the attack was conducted. Once the node is detected, it is identified to be the launch-pad for the attack and the real attacker lies somewhere else. Hence, at each step, an intermediate node or stepping stone has to be detected until the first node in the chain is found. It is evident that identifying each of the stepping stones in a long chain in the Internet is extremely difficult. Individual organizations or even Internet Service Providers (ISPs) may not be able to get the data (log files in intermediate nodes, timing of packets, size of packets, etc.) necessary for the detection due to the heterogeneous nature of the Internet with so many stakeholders involved. Hence most of the studies in this field restrict the scope to the detection of stepping stones within a single administrative domain.

The first significant study [38] in this field was published in 1995 by Staniford-Chen and Heberlein. They used content-based *thumbprints*, which

are summaries of contents similar to checksum to identify two different interactive connections with similar content. Content-based techniques lost importance as it became possible to encrypt content at each intermediate node. Since then, a lot of studies have been conducted which try to detect stepping stones in a chain based on timing as well as packet-size correlation. A lot of these studies consider random jitter and chaff deliberately inserted by the attacker in the traffic to make detection difficult.

Most of the papers in this domain have been published 10-15 years ago. This raises a question whether stepping stone attacks have become irrelevant. In the following sub-sections we present a few counter-arguments. We present cases of advanced persistent threats (APTs) and APT-type attacks where the attacker has access to a part of the network where he stays for a long time in order to steal data.

3.1.1 Panama Paper Leak - A Case Study

In the Panama Papers Leak incident [6, 36, 40] in April 2016, attackers gained access to the email server of a Panama-based firm. According to the speculations, the external attacker exploited vulnerabilities in the email server to compromise it. The attacker then exploited this compromised server as a stepping stone to gain more knowledge of the internal network and steal highly confidential documents revealing client information. This attack is an example of APT where the attacker has spent a long time in the internal network of the company undetected, interactively exploring the network and eventually stealing terabytes of data. The nature of the attack is similar to stepping stone attacks where attack traffic as well as stolen data pass through a chain of stepping stones before reaching the victim and the attacker respectively.

3.1.2 Surveillance Video Stream Hijacking

In Figure 3.1, digital video cameras send the video streams to a storage device. An external attacker might gain access to the camera by exploiting a vulnerability after which he can forward the video streams to an arbitrary location in the Internet. Here, a compromised host in the intranet acts as a launch pad or stepping stone. Videos can be delta compressed and only the changes from one frame to the next are forwarded to the storage device. In this case, the network traffic might have inherent ON (data) and OFF (no data) periods. The attacker may also want to watch a live stream and only enable it intermittently.

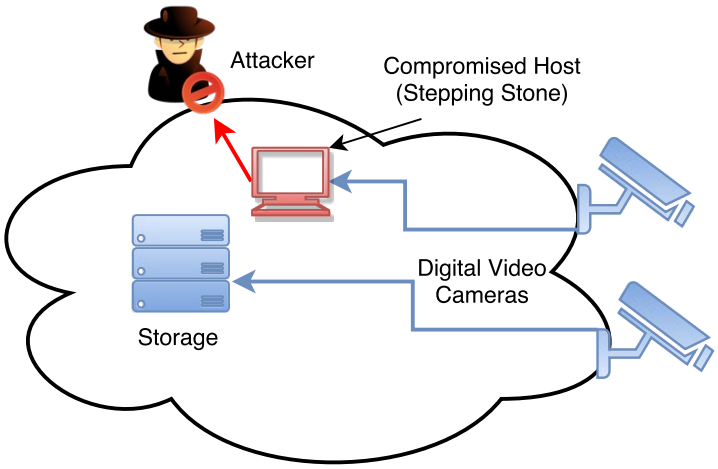

Figure 3.1: Video stream hijacking

3.1.3 Anonymous Networks and Tor

The aim of anonymous networks like Tor [41] is to provide anonymity to the users. Generally, such a network consists of a set of relay servers operated by volunteers. In the Tor network[1], the traffic between the client and each relay node is symmetrically encrypted using keys generated through authenticated key exchange protocols. There are three relay nodes in a path and each node decrypts the top layer of encryption and forwards the decrypted content to the next node in the path. The last relay node in the path removes the last layer of encryption and forwards data to the destination. This layered encryption scheme helps users to attain privacy and security as well as evade censorship. The anonymity is directly related to the number of users in the network and increases with the size of the population.

The Tor client is a free software and it is used to fetch the list of available relay nodes from the directory server and to select a random path to the destination through the Tor network. Hence attackers can use this anonymization network to launch attacks and hide behind the relay nodes. The intermediate nodes can reside within the target network or outside it. Figure 3.2 depicts the scenario where an attacker, in order to hide his identity, channels the attack traffic through one gateway followed by one or more external stepping stones back to the network through another gateway before attacking the victim. In this case, it is important to correlate the connections carrying attack traffic in order to identify and isolate the attacker and the stepping

[1]https://www.torproject.org/about/overview.html.en

stones.

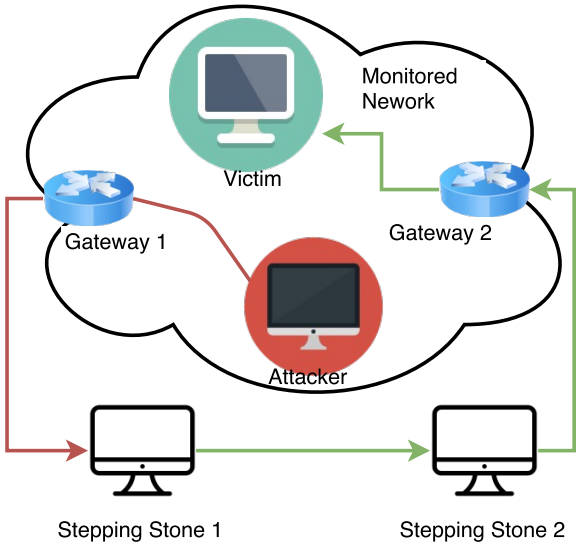

Figure 3.2: Hiding identity behind an external stepping stone

3.1.4 Botnets as Stepping Stones

A botnet is a collection of hosts, which are typically geographically distributed, under the control of a hacker, and used mainly for malicious purposes. Hosts become infected by malware like worms, Trojans or rootkits that turn them into bots. Generally, the bot client is downloaded to the host by Trojans or rootkits. The bot client communicates with one or more command and control (C&C) servers. This communication is sometimes preceded by a DNS resolution phase. The C&C layer lies between the attacker and the bots and is used to hide the attacker identity. This layer is responsible for relaying the commands from the attacker to the bots using protocols like Internet Relay Chat (IRC) [22] and HTTP. Peer-to-peer botnets use protocols like Kademlia [26] to control the bots. These bots are used to launch attacks such as distributed denial of service, spamming and port scanning.

Botnet-as-a-Service (BaaS) [4] is a new criminal service model which enables attackers to rent a botnet or a subset of it from the botnet controller. The attacker may launch attacks using the rented botnet as a launchpad. The C&C proxy layer hides the identity of the attackers even if the botnet is detected and taken down.

3.2 Detection Techniques

Various studies have been conducted on how to effectively detect stepping stones. As the attackers started using more advanced techniques like botnets and anonymity networks, studies continued to identify bots and deanonymize the attacker. Stepping stones can exist in legitimate computer systems and people use stepping stones for regular activities. Studies show that some of the existing detection techniques can erroneously identify legitimate Voice over IP (VoIP) traffic [43] and gateways [15] as stepping stones. It is important to reduce these and other false-positive cases. It is also important to identify legitimate and attack stepping stones and to analyze their traffic patterns and other properties.

3.2.1 Content-Based Detection

Early studies [38, 46] focussed on analyzing the payload of packets to detect stepping stones. Staniford-Chen and Heberlein proposed a *thumbprint* based solution [38]. The thumbprints are very short summaries of contents over a certain period in a connection, which are generated and stored at individual nodes within the network. When an intrusion is detected, these thumbprints are used to correlate connections and identify the chain of stepping stones. The authors also identified the properties which thumbprints should have. The effectiveness of the content-based detection techniques is limited by the fact that connections can be encrypted between the intermediate stepping stones.

3.2.2 Transmission Characteristic-Based Detection

Stepping stone detection techniques may rely on timing and size of packets in different connections. Timing based approaches use various parameters like inter-packet delay (IPD) and round-trip time (RTT).

Timing-based detection techniques Zhang and Paxson [55] proposed a detection technique which relies on packet size and timing to correlate connections in a chain of stepping stones in order to identify the stepping stones. The timing-based algorithm tries to identify ON (data) and OFF (no data) periods in a connection and is motivated by the spacing between human keystrokes in an interactive terminal, which follows Pareto distribution. If there is no data for time T_{idle}, then it signifies the onset of an OFF period. An OFF period ends when the next data packet arrives. If the ending times

of two OFF periods in two different connections differ by $\leq \delta$ seconds, they are said to be correlated. A constraint here is that the sink of one connection should be the source of the other connection. Two different connections are said to be correlated if

$$\frac{OFF_{1,2}}{min(OFF_1, OFF_2)} \geq \gamma$$

where $OFF_{1,2}$ denotes the number of correlated OFF period endings, OFF_1 denotes the number of OFF periods of the first connection, OFF_2 denotes the number of OFF periods of the second connection, and γ is a control parameter. The study takes into account the *causality constraint* according to which a packet can leave a node only after it arrives at the node. Some other refinements relate to the consideration of consecutive correlated OFF periods and reduce the number of false positives. The approach is unable to distinguish legitimate stepping stones from ones used for attacks, which results in a lot of false positives.

Yang and Huang [52] proposed a detection technique based on the analysis of the RTT of connections. As the length of a chain of stepping stones increases, the RTT increases following a step function. Hence, the length of the chain can be estimated by analyzing various RTT values. Other similar detection techniques [39, 48] use principal component analysis, neural networks, etc.

Packet count-based detection techniques He and Tong [20] proposed a detection technique which does not depend on timing. They assumed that the memory in any host is bounded, the timing delays are bounded and the packets are ordered. They used a counting-based algorithm with linear time complexity to detect stepping stones.

Thumbprinting Yang and Huang [51] proposed the idea of using temporal thumbprints in detecting stepping stones. Temporal thumbprints or T-thumbprints are sequences of temporal gaps between adjacent packets in an interactive TCP connection. The real-time algorithm tries to correlate T-thumbprints in order to identify consecutive connection pairs in a chain of stepping stones.

Watermarking Wang et al. [46] proposed an active stepping stone detection technique called Sleepy Watermark Tracing. When an intrusion is detected, watermarks are injected into the backward connection and collaboration with routers along the chain of stepping stones leads to the identification of the source of the chain. The technique does not use resources when

no intrusion is detected. It can detect stepping stones even if no data is transferred through the chain of connections. The watermarking technique was later used by several other proposals [33, 35, 44].

Anomaly-based detection techniques Crescenzo et al. [11] argued that active injection of jitter and chaff may decrease the chance of stepping stone detection with the timing-based algorithm proposed by Zhang and Paxson [55] (see page 26). Jitter or delay of more than δ, when introduced in at least one of the stepping stones, prevents the algorithm from correlating OFF periods. Typically δ is a time-span of a few milliseconds. Deliberate injection of chaff packets, on the other hand, reduces the value of γ thus making the algorithm ineffective. Hence, the algorithm should be complemented by three anomaly detection techniques. Naive stepping stone attacks are detected by the timing-based algorithm, while anomaly detection techniques identify connections with jitter or chaff as anomalous.

Response-time based anomaly detection uses the fact that a packet in the forward direction of a connection should be followed by a packet in the reverse direction within some time window. The method marks connections as anomalous (due to jitter) when they do not follow this principle. *Edit-distance based* anomaly detection builds on the idea that the sequence of ON and OFF periods in the forward direction of a connection should be similar to the sequence in the backward direction and have low edit-distance values. Injecting chaff packets results in the increase of this distance value and leads to the identification of anomalous connections. *Causality based* anomaly detection is based on the idea that, in a normal interactive connection, every pair of consecutive ON periods in the forward direction of the connection is associated with exactly one ON period in the backward direction, and vice versa. This detection technique can identify connections with chaff as anomalous.

3.2.3 Deanonymization Techniques

There are ways to deanonymize anonymous network traffic. For example, in case of Tor networks, if an observer can view the traffic on the first link (between user and the first Tor relay) and on the last link (between the Tor exit node and the destination), the traffic can be correlated based on timing. [2]

[2] https://www.torproject.org/docs/faq.html.en

3.2.4 Novel Techniques for Botnet Detection

Deep packet inspection Deep packet inspection includes header scanning, payload scanning, knowledge of various protocol (IRC, HTTP, etc.) semantics and classification based on this knowledge. BotHunter [17] identifies bots by mapping the activities to various stages observed in a bot life-cycle. BotSniffer [18] aims to identify communication of bots with C&C servers. Tools exist to classify network applications and map malicious activities in order to detect bots.

Scanning traffic Botnets with peer-to-peer communications for C&C may be detected using tools like BotMiner [16], which rely on correlating peer-to-peer communication with malicious activities. The tool identifies clusters of hosts with similar peer-to-peer communication and clusters from activities like port scanning and spamming. These clusters are then correlated to identify botnets. The tool uses Snort [37], an intrusion detection system, to detect the malicious activities.

DNS based detection There are various DNS based botnet detection techniques of varying complexity. Villamarín-Salomón and Brustoloni [42] used Bayesian probability theory to identify bots of the same botnet by analyzing the DNS queries they make over time. Choi et al. [7] identified bots by clustering hosts based on similarities in their DNS queries. Yadav et al. [50] focused on detecting domain flux techniques used by various botnets. Domain flux is a technique to dynamically generate domain names that identify C&C servers or proxies. The domain names can range from random alphanumeric strings to dictionary words. In the former case, the detection techniques rely on the fact that the distribution of alphanumeric characters are different in randomly generated strings and normal domain names. In the latter case, multiple metrics are required. Another study [12] shows ways to detect C&C communication that is tunnelled through DNS messages.

Spam-bot detection Botnets are often used to send spam emails, and this property can be used to identify the bots. A study by Xie et al. [49] focuses on identifying URLs in spam emails. Obfuscated URLs are detected by regular expression validators. Another study [14] aims to identify C&C servers after detecting the spamming bots. This study tries to model legitimate emails and spam emails and identify spams based on the distance to these models.

Communication analysis A study [23] on communication analysis for botnet detection relies on the ports to which individual hosts connect, fan-in

patterns, flow models of IRC and HTTP communications, and periodicity of communications. Another study [28] analyzes random walks in communication graphs and is mostly concerned with P2P C&C communication.

Detection using SDN A study [47] in this area has proposed a botnet detection technique specific to the architecture of SDN and the separation of data plane and control plane. The botnet detection components consist of generic templates, flow collector, multistage filtering, bot detection engine and attack prevention. The system uses IPFIX and customized templates for capturing useful flow information at the switches. The flow collector uses customized storage templates for storing the flow records reported by the switches. The multistage filtering is a five-stage process that filters out information related to normal traffic. The botnet detection engine uses various machine learning techniques in order to identify botnets with varying communication patterns. Both spatial and temporal communication patterns are taken into consideration to detect bots as well as botnets. The attack prevention component isolates an identified bot by configuring access control policies in the OpenFlow switch.

3.2.5 Legitimate Stepping Stone Detection

Users sometime use stepping stones legitimately for various activities. Not all chains of stepping stones are created with malicious intent. These cases need to be filtered out in stepping stone detection mechanisms which trigger a response, such as isolation of the stepping stones or the first node in the chain. A relevant study [10] has proposed an anomaly-based legitimate stepping stone connection detection technique to be used in conjunction with the prevalent timing-based detection techniques in order to reduce the false positive rate. The study uses a component that stores information regarding normal behaviour and provides reference data to the anomaly detection component. The study fails to document further details regarding the reference data and what information might be useful to construct such reference data.

Chapter 4

Timing-Based Detection in SDN and NFV

SDN and NFV are technologies that will be heavily used in future communication networks, and it is important to enable techniques for monitoring them. As discussed in Chapter 2, the architecture is different from that of traditional networks. The data plane and the control plane of an SDN are separated, and the network relies heavily on virtualized network functions. Stepping stone attacks are also possible in these new environments. The packets between consecutive stepping stones flow through the switches in the data plane programmed by the controller.

Challenges The challenge is that, in SDN, the often used south-bound protocols (e.g. OpenFlow [3]) are not suitable for traffic monitoring. The controller can gather flow-level statistics using these protocols but cannot gather useful monitoring information on individual connections. Instead, one has to use protocols such as NetFlow/IPFIX and sFlow to gain detailed knowledge about the traffic passing through the individual switches in the data plane. As discussed already in Chapter 2, sFlow is more scalable than the other alternatives. Switches can be configured to sample header information at a specific rate and to forward that information to a collector. Regarding stepping stone detection, the challenges include removing redundancy from the collected data, identifying connections and ON/OFF periods of those connections from the sampled data, and correlating connections based on the collected information.

In this chapter, we explain the packet sampling and timing-based detection of stepping stones. We also set the evaluation goals of our experiments and present the implementation details. We also discuss about the various

network topologies which we have considered in our experiments and the sFlow security model in general.

4.1 Packet Sampling and Timing-Based Detection

We rely on the switches in the data plane for sampling packets and forwarding header information to a central collection and analysis module. The switches should be able to operate at varying packet sampling rates. We aim at real-time identification of stepping stones and, hence, it is important for the switches to immediately forward the sampled information to the analysis module. Any delay will have considerable impact on the detection procedure. Therefore, a packet sampling and reporting mechanism with no caching and delay is preferred.

4.1.1 Timing-Based Detection

The sampled header information is analyzed to identify stepping stones by correlating connections. First the ON and OFF periods of connections are identified. Then these periods of different connections are correlated. The connections should be consecutive, i.e., the source of one connection should be the destination of another connection. Finally, consecutive connections are correlated based on the period correlation. The analyzer module uses the same sampled header information to identify connections that are anomalous due to jitter and chaff.

Identification of ON/OFF periods As discussed in Chapter 3, interactive connections can be structured into ON (data) and OFF (no data) periods based on keystroke spacing of the user which can be described by a Pareto distribution. It has been observed [32] that 25% of keystrokes are 500 milliseconds or more apart. Similar to the solution proposed by [55], we consider a connection to enter an OFF period when there is no data for T_{idle}. An OFF period ends and an ON period begins when the first data packet arrives after the onset of the OFF period. When inter-packet spacing is less than T_{idle}, each data packet contributes to the size of the content transferred in the corresponding ON period.

Correlating ON/OFF periods Zhang et al. [55] proposed that two OFF periods of two different connections are correlated if their ending times differ

by a value $\leq \delta$ milliseconds. As the connections are characterized by alternate ON and OFF periods, there is no difference in correlation based on the ending times of OFF periods or the starting times of ON periods. In our study, two ON periods of two different connections are correlated if their starting times differ by a value $\leq \delta$ seconds. While correlating ON periods, we order the pair of ON periods $\{a, b\}$ by a *happens after* relationship. In this case, b *happens after* a, that is, the ON period b starts no later than δ seconds after the onset of ON period a. We do not correlate ON periods of the forward and reverse legs of the same connection.

Correlating connections The *timing-based correlation score* of two connections is given by

$$\frac{ON_{1,2}}{min(ON_1, ON_2)}$$

where $ON_{1,2}$ denotes the number of correlated ON period starts, ON_1 denotes the number of ON periods of the first connection and ON_2 denotes the number of ON periods of the second connection. The first and second connections are ordered by the *happens after* relationship between their ON periods as discussed above. The *content-size based correlation score* is based on the idea that, for two correlated connections, if the content size increases from one ON period to the next of one connection, it would increase correspondingly for the other connection. For this purpose, we only consider those pairs of ON periods which are correlated. The score is given by

$$\frac{Matches\ found}{Number\ of\ correlated\ ON\ period\ pairs - 1}$$

Two connections are identified as correlated if both

$$timing\text{-}based\ correlation\ score \geq \gamma_{timing}$$

and

$$content\text{-}size\ based\ correlation\ score \geq \gamma_{content\text{-}size}$$

where γ_{timing} and $\gamma_{content\text{-}size}$ are tunable parameters.

Handling jitter and chaff We aim to adapt the anomaly-based jitter and chaff detection techniques proposed by Crescenzo et al. [11] to the environment under consideration. *Response-time based* anomaly detection, used for detecting deliberately inserted jitter in the attack traffic, does not work in a sampled environment as the technique relies on mapping each packet in the forward leg of a connection with its response in the reverse leg of the same

connection based on round trip times calculated using the Jacobson-Karel's algorithm [21]. In an environment that relies heavily on sampling, this mapping cannot be effectively performed. *Edit-distance based* anomaly detection and *causality based* anomaly detection, as discussed in Chapter 3, can be adapted to this environment because of the fact that these chaff detection techniques rely on analyzing ON and OFF periods in interactive connections rather than individual packets and their responses.

De-anonymization We aim to de-anonymize an attacker who uses an anonymity network to hide his identity. If the victim host and the attacker host both lie within the network while the anonymity network is external to the network, the first link and the last link in the chain of stepping stones can be monitored. If these two links can be correlated considering the section of intermediate nodes and links as a black box, the attacker can be de-anonymized. Hence we aim to correlate any arbitrary pair of connections and analyze the correlations. End-to-end deanonymization is more difficult than step-by-step correlation because jitter accumulates along the path, but it requires fewer sampling points in the network.

4.2 Evaluation Goals

Evaluation goals of our experiments are as follows:

- Estimate values of T_{idle} for which sufficient number of ON/OFF periods of a connection can be identified in different network topologies for varying packet sampling rates.
- Estimate values of δ for which ON/OFF periods as well as connections can be effectively correlated in different network topologies for varying packet sampling rates.
- Verify whether the γ_{timing} value of 0.3 (as estimated by Zhang et al. [55]) is correct for identifying correlated connections.
- Estimate $\gamma_{content\text{-}size}$ value for effective content-size based correlation.
- Analyze the effectiveness of edit-distance based and causality-based chaff detection techniques in different network topologies for varying packet sampling rates.
- Evaluate the possibility of de-anonymization by correlating arbitrary connection pairs in different network topologies for varying packet sampling rates.

4.3 Implementation Details

Emulation Environment We used Mininet[1] as the primary tool for emulating the necessary environment. Mininet is a popular tool for experimenting with SDN and OpenFlow and runs virtual hosts, switches and links on top of the same operating system kernel. Although the components are created with software rather than real hardware, real network behaviour can be replicated in this virtual environment. It is possible to create different network topologies and to *ssh* [53] to the virtual hosts. The virtual switches support OpenFlow as well as sFlow protocols.

sFlow packet sampling fits our implementation as the switches immediately forward the sampled header information without caching. The sFlow sampling rate and polling rate can be configured for the switches. These rates can be adjusted to fit the type of the network (bandwidth, traffic volume, etc.) so that the sFlow collector is not flooded with sFlow datagrams. As different sampling rates are used, the probability that a packet traversing a specific switch will be sampled and reported to the collector varies. For example, if the sampling rate is 1 in s packets, then the probability of a particular packet getting sampled is *1/s*. Hence, if packets have to pass through n switches on average, the probability of a packet being sampled in at least one of the n switches is $1 - ((s-1)/s)^n$ which increases with n for a fixed value of s. On the other hand, for a fixed average number of switches n, the probability of a packet being sampled in at least one of the switches decreases with increase in the sampling rate s.

There is also a related trade-off which should be considered while designing network monitoring systems for SDN environments. If the probability of a packet being sampled increases beyond a threshold, it may result in flooding at the sFlow collector and hence reduce the scalability of the system. Hence, monitoring strategies have to be devised which can gain sufficient insight from sampled data and make the monitoring system truly scalable.

Technologies used Mininet uses by default a reference controller which installs flow entries to the flow tables of the switches through the SBI. We require a framework to monitor the network in order to identify the stepping stones as OpenFlow counter values do not provide sufficient information. Hence, an sFlow controller is needed to gather traffic-related metadata from the switches (sFlow agents). We used Node.js[2] to implement the sFlow con-

[1] http://mininet.org/
[2] https://nodejs.org/en/

troller and MongoDB[3] as the forensic data store for traffic data.

In our study, we have used Mininet 2.2.1, Open vSwitch 2.0.2, OpenFlow 1.0 and sFlow 5.0. For the forensic data store we have used MongoDB 3.2.4.

Implementation As shown by Figure 4.1, the sFlow controller can be logically divided into two different modules: the sFlow collector and the data analyzer. The modules interact with the forensic data store to store and retrieve traffic data and to store correlation information. An application can retrieve information about correlations and stepping stones from the forensic data store.

We implemented two different versions of the data analyzer. The version which is discussed here detects stepping stones in real-time based on the sFlow datagrams forwarded by the sFlow capable switches in the network. Another version of our application analyses captured network traces, simulates sFlow sampling and identifies stepping stones.

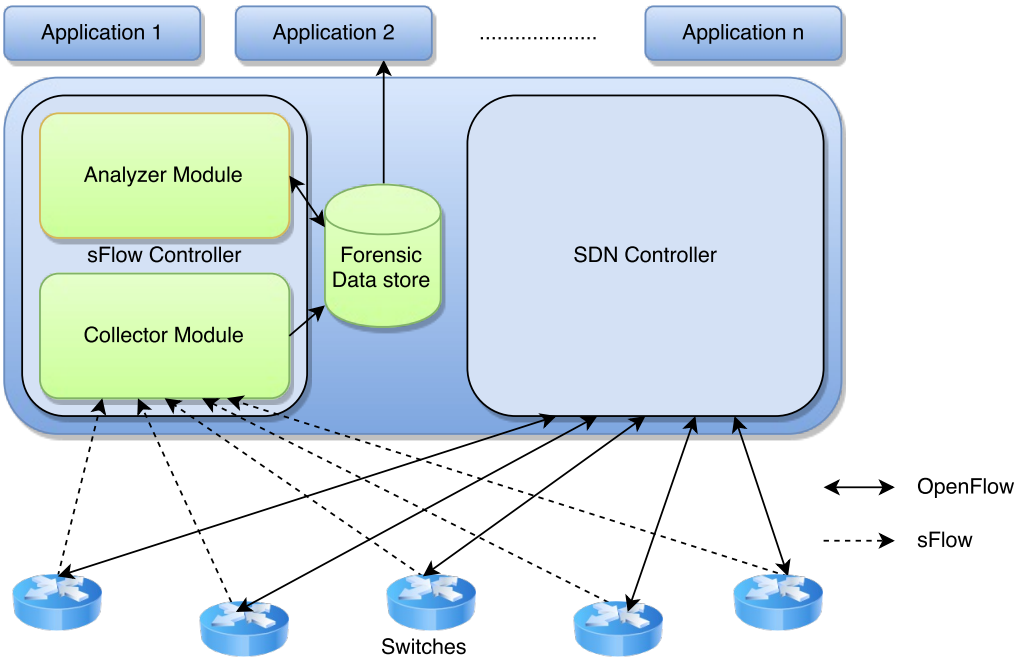

Figure 4.1: The architecture.

The sFlow agents embedded in switches can be configured to modify the sampling and polling rates. The sFlow agent extracts header information

[3]https://www.mongodb.org/

from the sampled packets, marshals the header information into sFlow datagrams and sends the datagrams immediately to the sFlow collector. The sFlow *collector module* extracts the header information from the sFlow datagrams and stores the information in a data-store to be used by the data-analysis module in order to correlate connections and detect stepping-stones.

The *data-analysis module* works on the traffic data collected by the data-collection module. It fetches the raw traffic data comprised of header information and removes redundancy introduced by multiple sFlow agents in the path of a packet sending the same header information. Then it tries to match these headers with existing connections between the hosts. Once the matching has been done, the header information is used to update the connection data. The individual connection information is used to correlate the connections. The knowledge gained by this module can be used by an intrusion prevention system (IPS) application, which may instruct the SDN controller to isolate stepping stones and to restrict traffic generated at these stepping stones.

The *forensic data store* stores the necessary information for forensic analysis of the data. Although the header information received by the collector module gets temporarily stored here, the data store gets rid of unnecessary data and stores only meaningful data like period correlation information and connection correlation information, which can act as evidence in forensic analysis.

4.4 Topologies and Sampling Rates

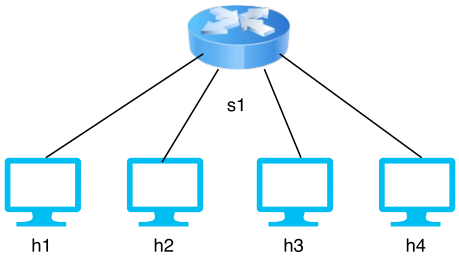

Figure 4.2: Single Switch or Star Topology

We consider four different topologies in our experiments: single switch or star topology, tree topology, linear topology and clos topology. The average number of switches between two hosts varies between the topologies. If the sFlow packet sampling rate is set to 1 in n packets and the average number of

switches between two end hosts is s, then the probability of a packet getting sampled in at least one of the switches is $1 - (\frac{n-1}{n})^s$.

In the single switch or star topology of Figure 4.2, every packet has to traverse a single switch before it can get delivered to the destination host. Hence, if the sFlow packet sampling rate is 1 in n packets, the probability of an individual packet getting sampled is $1/n$.

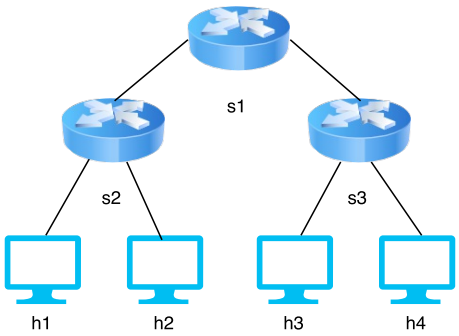

Figure 4.3: Tree Topology

In the tree topology of Figure 4.3, we have used *h1*, *h3*, *h2* and *h4* as the consecutive hops in our experiments. Hence, every packet has to traverse through 3 switches before it gets delivered to the destination host. If the sFlow packet sampling rate is 1 in n packets, the probability of a packet getting sampled in at least one of the 3 switches is $1 - (\frac{n-1}{n})^3$.

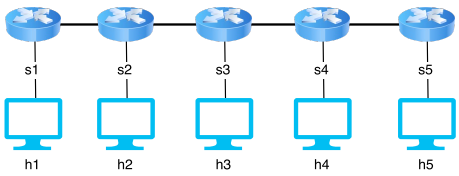

Figure 4.4: Linear Topology

In the linear topology of Figure 4.4, if host *h1* sends a packet to host *h5*, the packet traverses through 5 switches and hence the probability of it getting sampled is $1 - (\frac{n-1}{n})^5$ when the packet sampling rate is set to 1 in n packets.

In the clos topology of Figure 4.5, there are 5 switches between host *h1* and *h7* along any of the possible paths. Hence, following the same argument

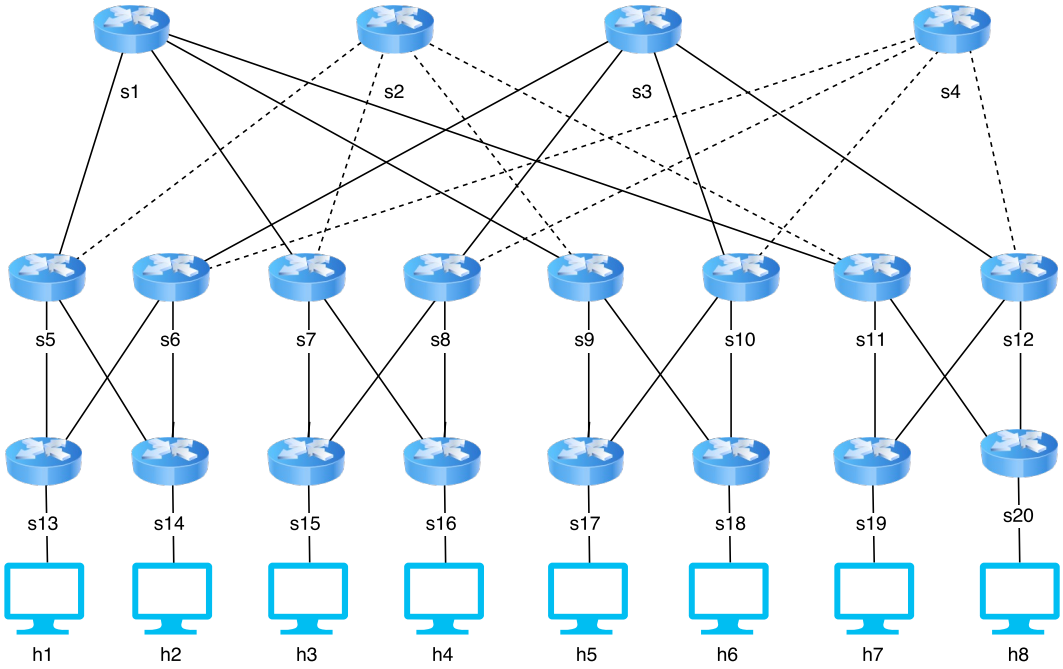

Figure 4.5: Clos Topology

as in the case of linear topology, the probability of a packet sent from *h1* to *h7* getting sampled is $1 - (\frac{n-1}{n})^5$ when the packet sampling rate is set to 1 in n packets.

4.5 sFlow Security

The deployment of network monitoring raises a number of security issues which need to be addressed. sFlow does not have any security mechanism and relies on proper deployment and configuration.

sFlow traffic is sent unencrypted to the collector. Casual eavesdropping as well as spoofing of datagrams are possible. To eliminate these issues, sFlow datagrams should be sent through an isolated channel. VLAN or VPN tunnels can be used to create these secure isolated channels. The solution is deployment specific and in our experiments we have simply forwarded the unencrypted traffic through the network to the collector. However, in our implementation, we check the sequence numbers of the headers encapsulated in the sFlow datagrams to remove possible redundancy or spoofed packets.

Analysis of the sFlow datagrams can reveal sensitive information about

the network activities of a user. Although sampling of packets at the switches and limiting the number of header bytes encapsulated by the sFlow datagram prevents leakage of sensitive information to some extent, only the network administrators with proper rights should be allowed to access the forensic data store for forensic analysis. Nevertheless, network monitoring itself makes the network more robust and less vulnerable to attacks.

Chapter 5

Results

In this chapter we present the results gathered while experimenting with stepping stone detection in different network topologies and varying sFlow packet sampling rates. Each experiment is run 100 times and the graphs plot results with 3 bars. The middle bar represents the average while the upper and lower bars depict the range in 95% of the cases.

5.1 Identification of ON/OFF Periods

From Figures 5.1, 5.2, 5.3 and 5.4, it is evident that the number of ON periods for a specific volume of traffic decreases with an increase in T_{idle} in the different topologies. Nevertheless, a T_{idle} value of 500 milliseconds leads to the identification of sufficient number of ON periods in the tree, linear and clos topologies. Even though the number of ON periods for T_{idle} set to 500 milliseconds is less than 10 in this specific case of the star topology, the value should be high enough for effective analysis of interactions done over considerable periods of time.

It is also evident that the number of ON periods increases with the increase in the number of sampling switches between two stepping-stone hosts. In our experiments, the packet sampling rate was set to 1/10. For packet sampling rate of 1/1, with the same generated traffic, 15 ON periods were identified when T_{idle} was set to 500 milliseconds. We attribute the increase in the number of ON periods for packet sampling rate of 1/10 in most of the topologies to the fact that some packets within a longer ON period were not sampled by any of the switches, which divided the single ON period into multiple smaller ON and OFF periods.

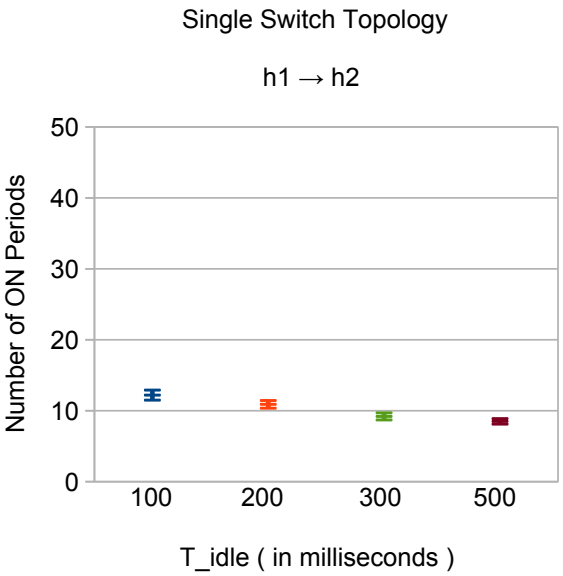

Figure 5.1: Effect of varying T_{idle} on the number of ON periods of connections for single switch or star topology. Packet sampling rate is 1/10.

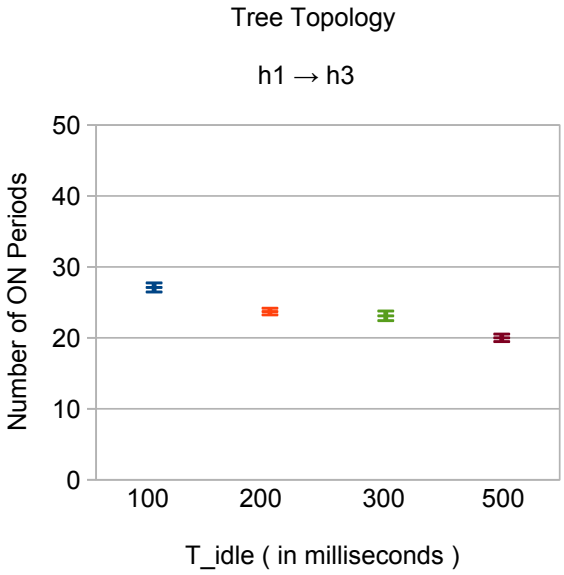

Figure 5.2: Effect of varying T_{idle} on the number of ON periods of connections for tree topology. Packet sampling rate is 1/10.

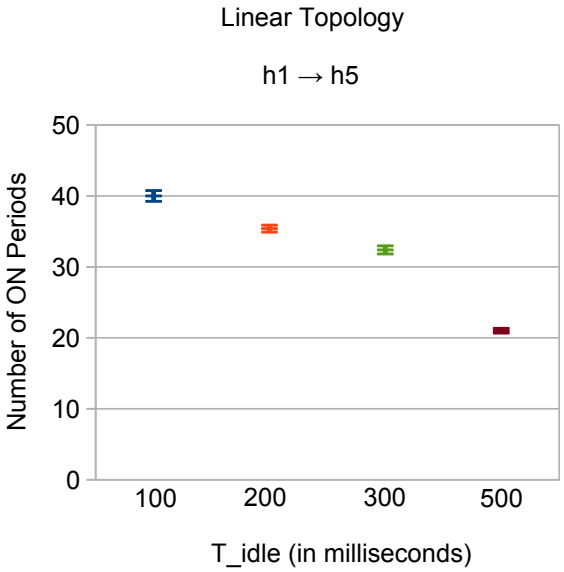

Figure 5.3: Effect of varying T_{idle} on the number of ON periods of connections for linear topology. Packet sampling rate is 1/10.

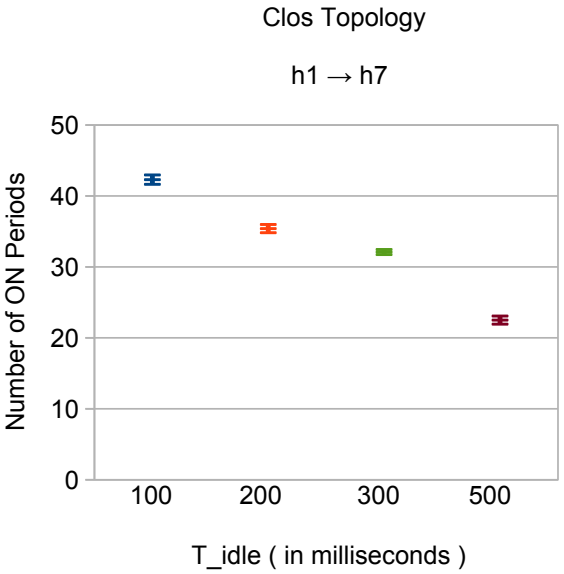

Figure 5.4: Effect of varying T_{idle} on the number of ON periods of connections for clos topology. Packet sampling rate is 1/10.

5.2 Correlation of ON Periods

In Figures 5.5 and 5.6, we present the observed number of correlated ON periods between two consecutive connections in a chain of stepping stones in a tree topology. Here the attacker hops from host *h1* to *h2* and then to *h3* and generates attack traffic. The number of correlated ON periods in consecutive connections $h1 \rightarrow h2$ and $h2 \rightarrow h3$ increases with the increase in δ. In the non-sampled case, low values of δ in the range of 1-3 milliseconds are sufficient to correlate ON periods. With a packet sampling rate of 1/10, a δ of 100 milliseconds is necessary to identify correlations between ON periods. Nevertheless, higher values of δ should be avoided in order to keep the false positive rate low.

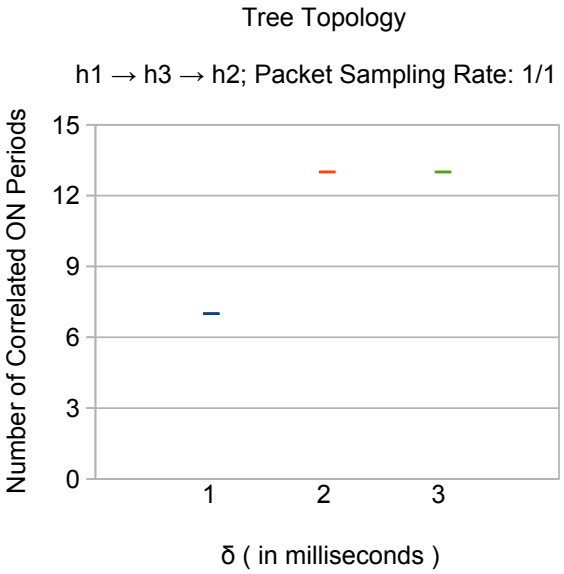

Figure 5.5: Effect of varying δ on the correlation of ON periods in a tree topology. T_{idle} is 500 milliseconds.

In Figures 5.7 and 5.8, we observe the number of correlated ON periods for attack traffic trace generated in a clos topology. Here the attacker hops from host *h1* to *h7* and then to *h3*. From the graphs, we can observe that the number of correlations of ON periods between consecutive connections $h1 \rightarrow h7$ and $h7 \rightarrow h3$ increases with the value of δ. When packet sampling rate is 1/1, the correlation is quite high for very low values of δ similar to the tree topology. Also, when packet sampling rate is set to 1/10, the number of

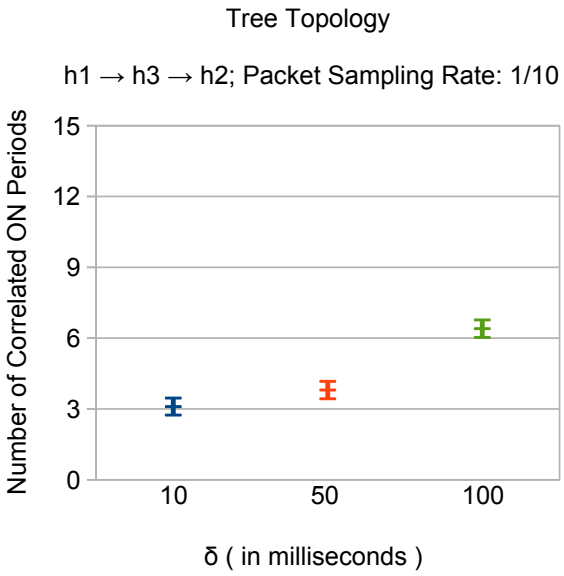

Figure 5.6: Effect of varying δ on the correlation of ON periods in a tree topology. T_{idle} is 500 milliseconds.

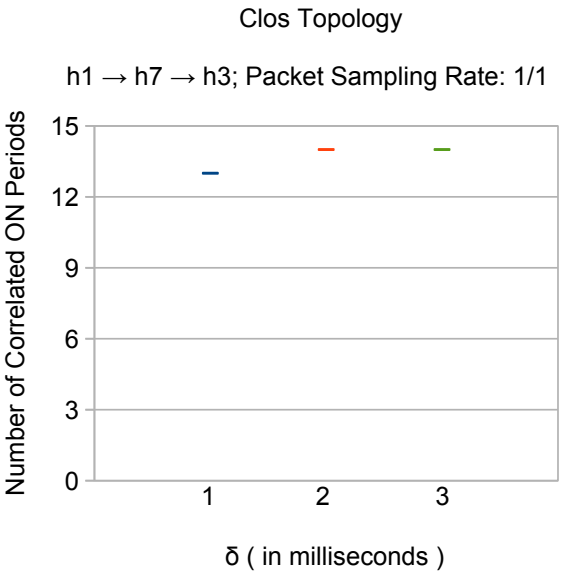

Figure 5.7: Effect of varying δ on the correlation of ON periods in a clos topology. T_{idle} is 500 milliseconds.

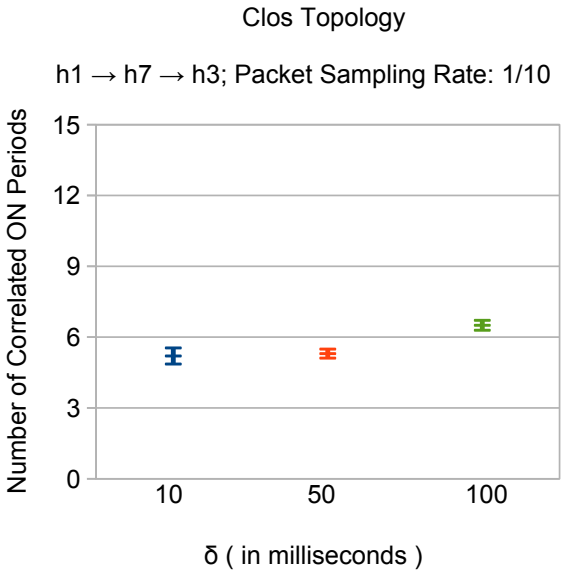

Figure 5.8: Effect of varying δ on the correlation of ON periods in a clos topology. T_{idle} is 500 milliseconds.

correlations is high for δ value of 100 milliseconds. Hence, the observations are similar for the tree and clos topologies and can provide us with possible values of δ to be used in different topologies and sampling rates.

5.3 Timing Based Correlation

The timing-based correlation score of two consecutive connections in a stepping stone chain in a tree topology increases with δ as depicted in Figures 5.9 and 5.10. The attacker hops from host *h1* to *h2* and then to *h3*. The timing-based correlation scores for the two consecutive connections $h1 \rightarrow h2$ and $h2 \rightarrow h3$ in the non-sampled case are higher than the threshold γ_{timing} value of 0.3 for δ in the range of 1 to 3 milliseconds. 1 being the highest possible value, the correlation score saturates for δ value of 2 milliseconds at a high value of 0.867. For packet sampling rate of 1/10, the scores are much lower even for much higher δ values but increase monotonically with δ. For δ set to 100 milliseconds or more, the correlation score lies above γ_{timing} and the connections are correlated.

From Figures 5.11 and 5.12, it is evident that running similar experiments in the clos topology results in observations similar to the tree topology. The

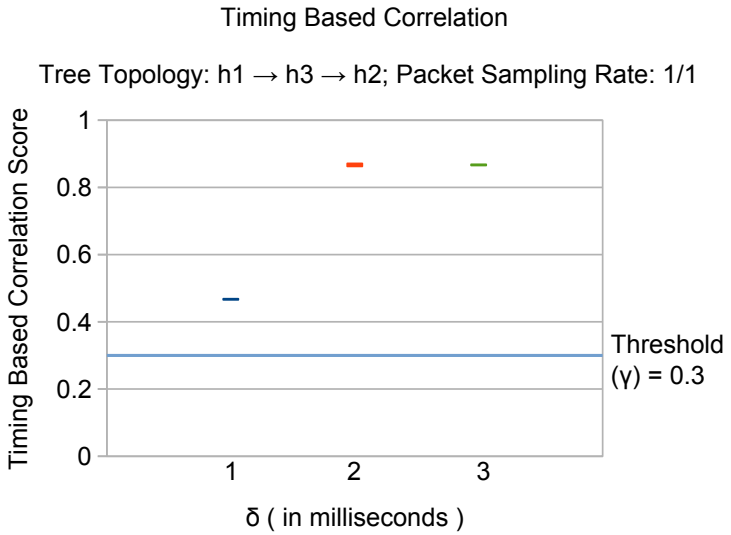

Figure 5.9: Effect of varying δ on the timing based correlation score in a tree topology. T_{idle} is 500 milliseconds.

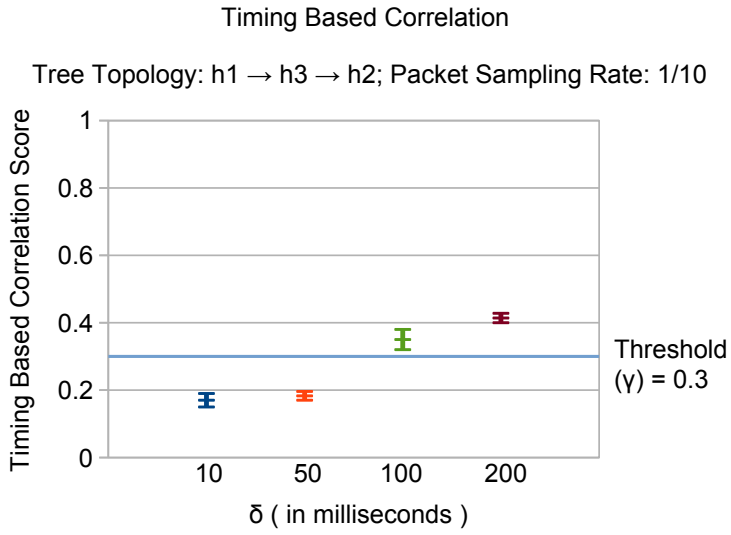

Figure 5.10: Effect of varying δ on the timing based correlation score in a tree topology. T_{idle} is 500 milliseconds.

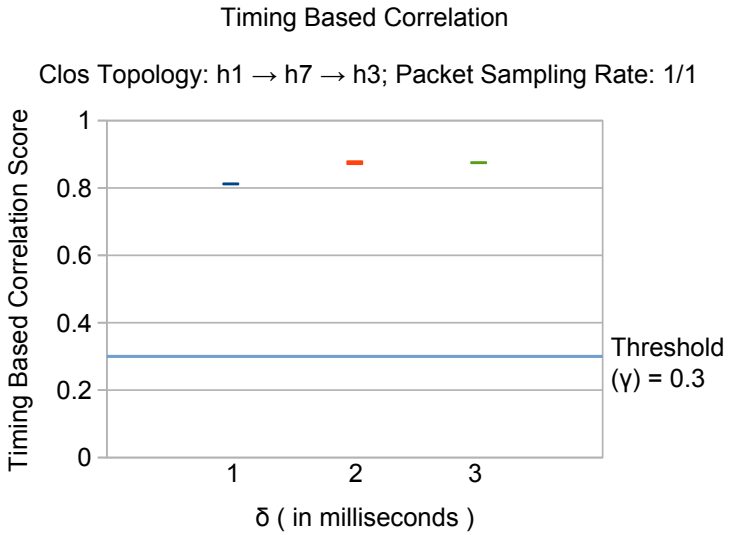

Figure 5.11: Effect of varying δ on the timing based correlation score in a clos topology. T_{idle} is 500 milliseconds.

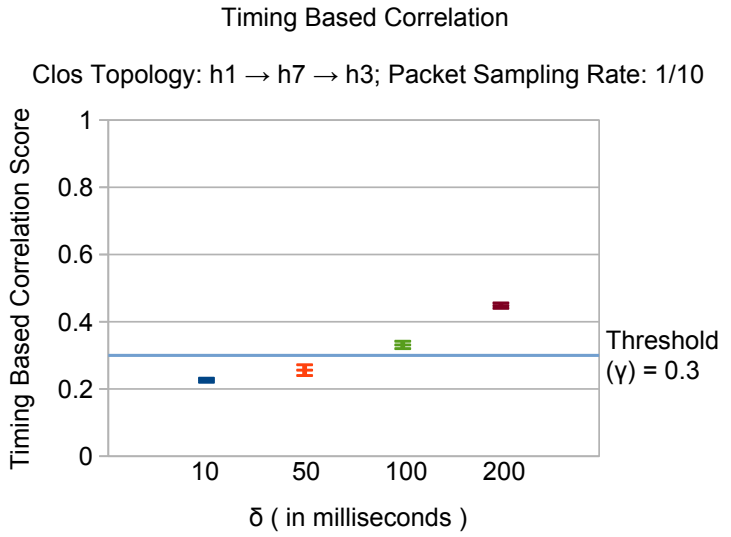

Figure 5.12: Effect of varying δ on the timing based correlation score in a clos topology. T_{idle} is 500 milliseconds.

timing-based correlation score is above the threshold value (γ_{timing}) of 0.3 for very low values of δ in the range of 1-3 milliseconds when packet sampling rate is 1/1. Hence correlated connections could be effectively identified. When sFlow packet sampling rate is set to 1/10, effective correlation is possible for δ values of 100 milliseconds and higher. We should keep in mind that very high values of δ, on the other hand, leads to increase in the rate of false positives and should be avoided.

5.4 Content-Size Based Correlation

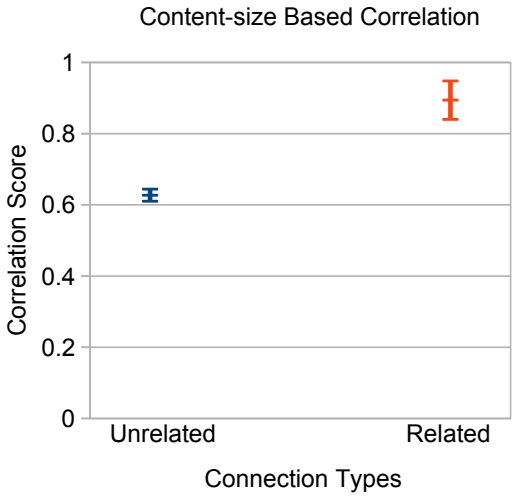

Figure 5.13: Content-size based correlation score for related and unrelated connections.

Content-size based correlation is used as a second check in order to reduce the number of false positives. Even if two unrelated connections generate a very high timing based correlation score, it is unlikely that the content sizes over the ON periods would show a similar pattern for the connections. Figure 5.13 presents the graph which depicts that this additional measure is effective and the differences in scores between unrelated and related connections is high. In case of connections which are unrelated, the score lies near to the ideal value of 0.5 with an average score of 0.627 in the 95th percentile. In case of related connections the average content-size based correlation score is 0.894 which is near the ideal score of 1. These observations motivate us to set the threshold value $\gamma_{content\text{-}size}$ for content-size based correlation to

0.8. Connection pairs should be marked as correlated if and only if both the timing-based correlation score and the content-size based correlation score lie above the defined thresholds.

5.5 Edit-Distance Based Chaff Detection

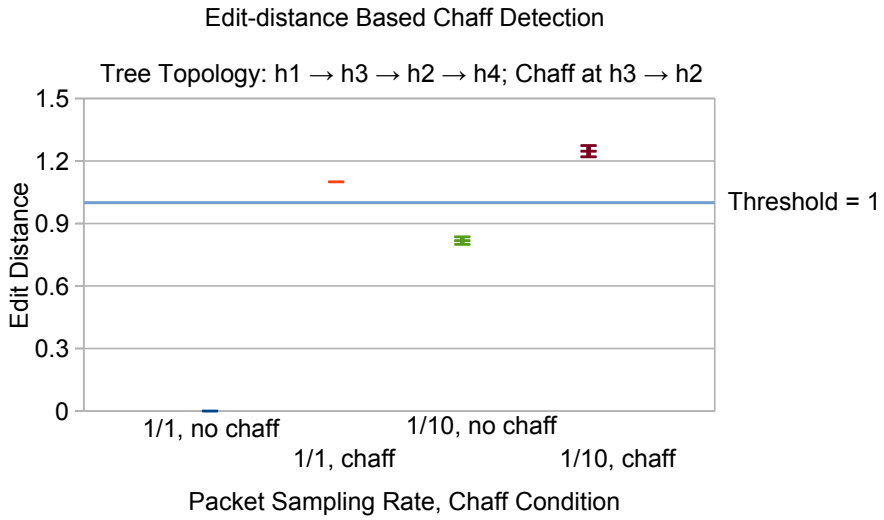

Figure 5.14: Edit distance based chaff detection in sampled and non-sampled environments in tree topology.

Edit-distance of an interactive connection is high if the attacker inserts chaff in the traffic. The threshold distance is 1 and a connection is marked as anomalous due to chaff if the observed edit-distance is more than this threshold. In our experiments, the attacker establishes an ssh session in a tree topology from host *h1* to *h4* using intermediate hosts *h3* and *h2* as the stepping stones. The attacker then introduces chaff in the connection $h3 \rightarrow h2$ in the stepping stone chain $h1 \rightarrow h3 \rightarrow h2 \rightarrow h4$. The same attack traffic trace is fed to our application for both sampled and non-sampled cases. We run our experiments for traffic with chaff and no chaff. As depicted in Figure 5.14, the detection technique is able to identify anomalous connections effectively in a tree topology. The difference between the edit distances when there is chaff and when there is no chaff is higher when packet sampling rate is 1/1. Nevertheless, even when the sFlow packet sampling rate is set to 1/10, the edit-distance lies above the threshold when there is chaff.

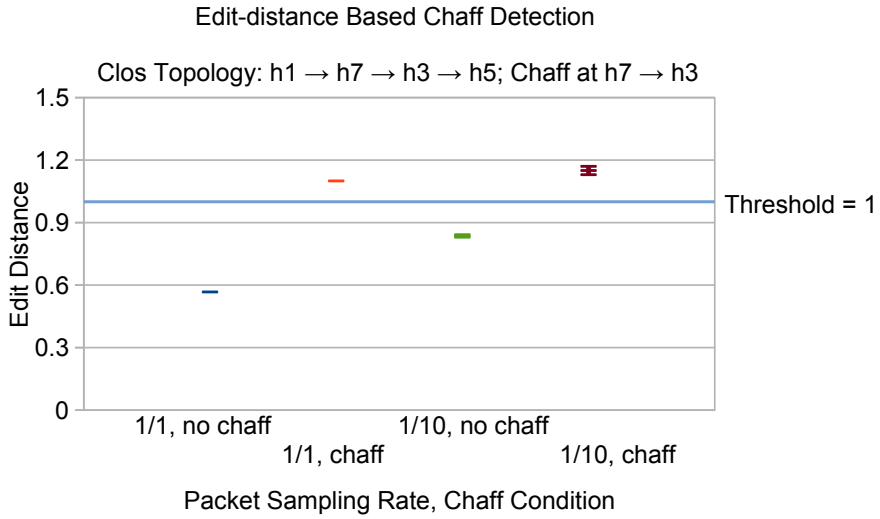

Figure 5.15: Edit distance based chaff detection in sampled and non-sampled environments in clos topology.

In clos topology (Figure 5.15), we ran similar experiments. Here the stepping stone chain is $h1 \rightarrow h7 \rightarrow h3 \rightarrow h5$ and chaff is deliberately introduced in the attack traffic at $h7 \rightarrow h3$. Observations are similar to those in the tree topology, and the edit distances for the connection $h7 \rightarrow h3$ lie above the threshold distance of 1 when there is chaff in the attack traffic. The difference of scores in case of chaff and no chaff is smaller when the sFlow packet sampling rate is set to 1/10 but the detection mechanism is still effective enough to mark connections with chaff as anomalous.

5.6 Causality Based Chaff Detection

Causality based anomaly score of an interactive connection is low if there is exactly one ON period in the reverse leg of the connection between two consecutive ON periods in the forward leg and vice versa. When the attacker inserts chaff in the attack traffic, the score increases and the connection is marked as anomalous once the score rises above the threshold value of 0.67. We executed the detection mechanism on captured attack traffic trace in the same way as discussed in the previous section on edit-distance based chaff detection. As evident from Figure 5.16 and 5.17, the detection technique is effective when the packet sampling rate is 1/1 and the connections with injected chaff are marked anomalous. When the sFlow packet sampling rate

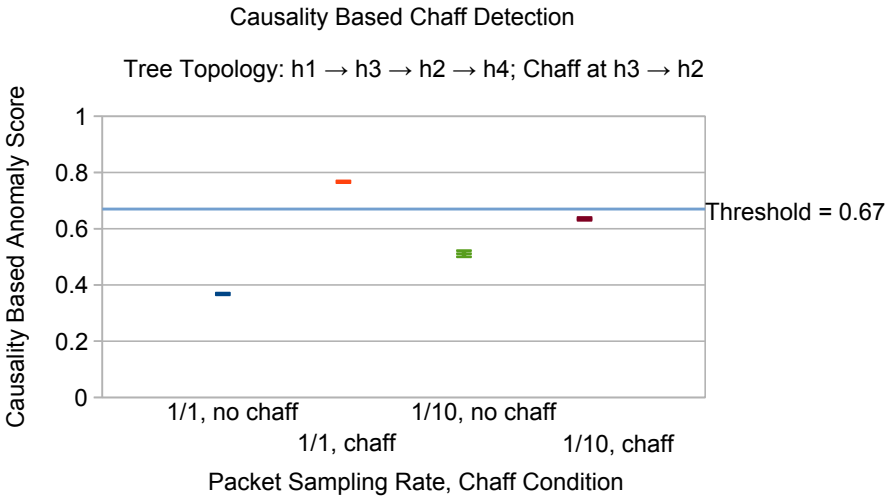

Figure 5.16: Causality based chaff detection in sampled and non-sampled environments in tree topology.

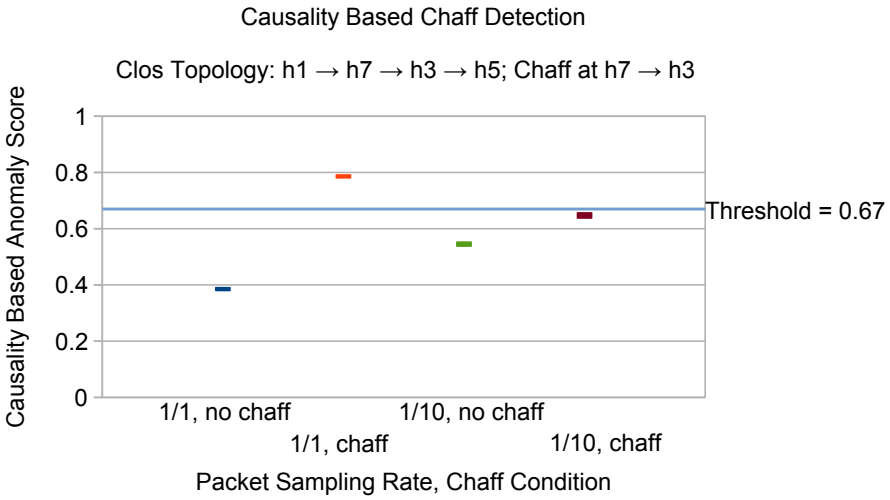

Figure 5.17: Causality based chaff detection in sampled and non-sampled environments in clos topology.

is 1/10, this detection technique is ineffective and the anomaly score lies below the threshold value even in positive cases. Hence, when lower packet sampling rates are enabled, edit-distance based chaff detection mechanism should be implemented in order to effectively detect anomalous connections with injected chaff.

5.7 De-anonymization

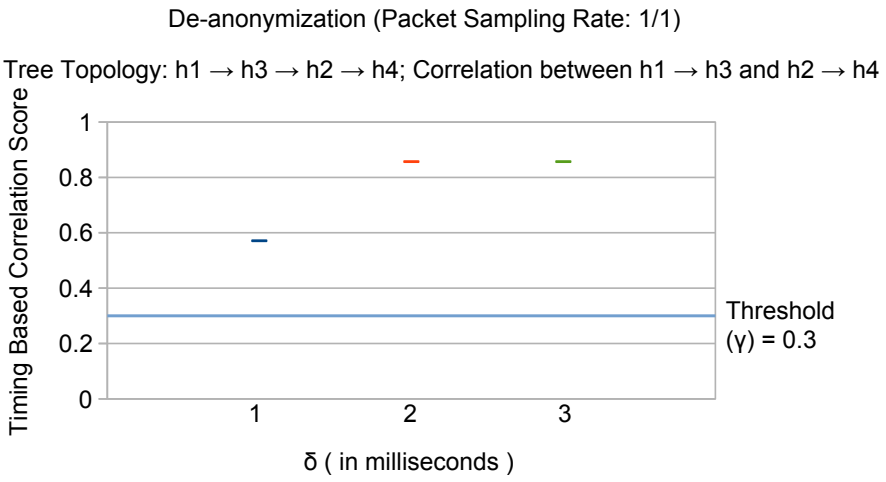

Figure 5.18: Effect of varying δ on the timing based correlation score of two non-consecutive connections in a stepping stone chain in a tree topology. T_{idle} is set to 500 milliseconds.

We tried to correlate first ($h1 \rightarrow h3$) and last connection ($h2 \rightarrow h4$) in a chain of stepping stones ($h1 \rightarrow h3 \rightarrow h2 \rightarrow h4$) in a tree topology. Figures 5.18 and 5.19 present the timing based correlation scores in the 95th percentile for the first and last connections when sFlow packet sampling is 1/10 and 1/1. When packet sampling rate is 1/1, the timing-based correlation scores are higher than the threshold value (γ_{timing}) of 0.3 for positive cases for very small values of δ in the range of 1-3 milliseconds. However, when packet sampling rate is 1/10, the timing-based correlation scores do not rise above the threshold value even for higher values of δ like 100 and 200 milliseconds.

In the clos topology, we tried to correlate the first and last links in a chain of connections $h1 \rightarrow h7 \rightarrow h3 \rightarrow h5$ in a similar way. When sFlow packet sampling rate is 1/1, the connections $h1 \rightarrow h7$ and $h3 \rightarrow h5$ are

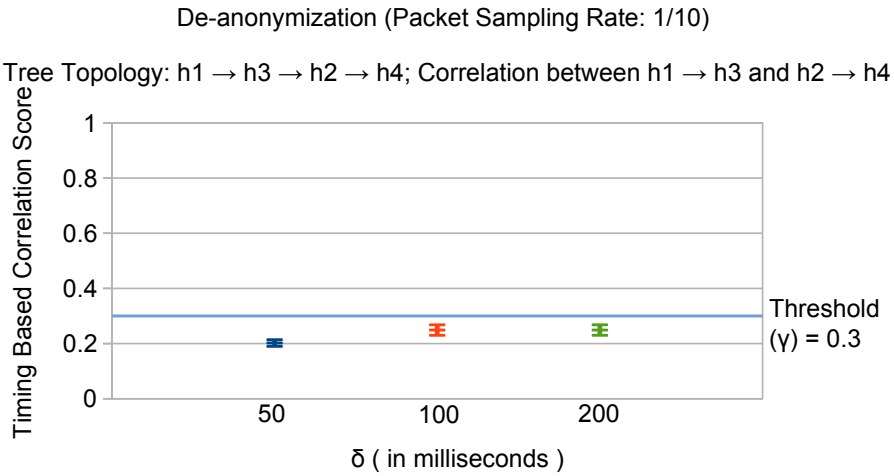

Figure 5.19: Effect of varying δ on the timing based correlation score of two non-consecutive connections in a stepping stone chain in a tree topology. T_{idle} is set to 500 milliseconds.

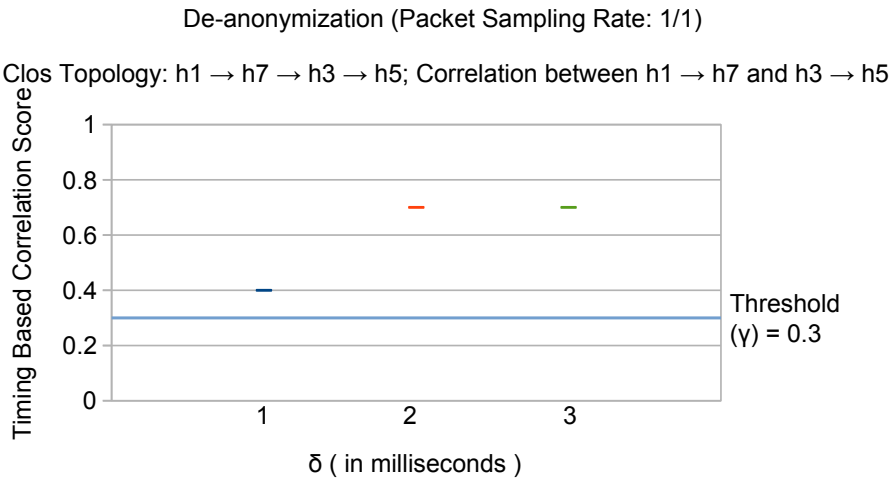

Figure 5.20: Effect of varying δ on the timing based correlation score of two non-consecutive connections in a stepping stone chain in a clos topology. T_{idle} is set to 500 milliseconds.

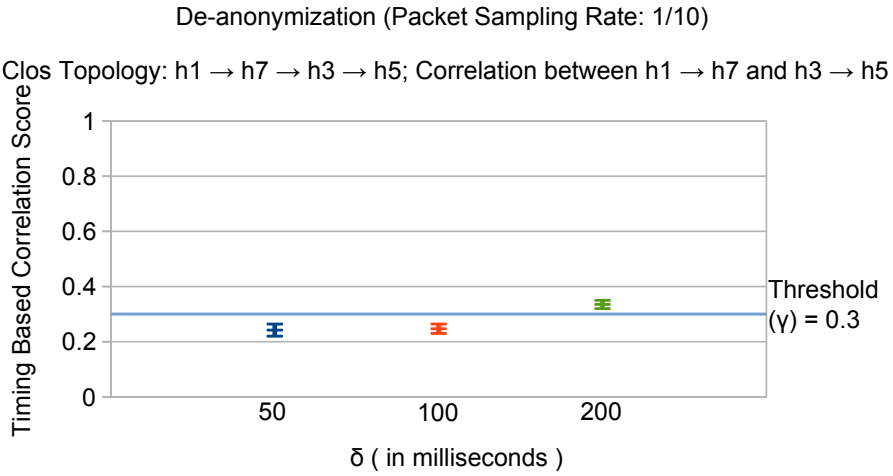

Figure 5.21: Effect of varying δ on the timing based correlation score of two non-consecutive connections in a stepping stone chain in a clos topology. T_{idle} is set to 500 milliseconds.

effectively correlated with timing-based correlation scores lying above the threshold value of 0.3 for low values of δ in the range of 1-3 milliseconds. When sFlow packet sampling rate is 1/10, the timing-based correlation score lies above the threshold value of 0.3 for δ set to 200 milliseconds. Hence, for high values of δ, we can effectively correlate a pair of non-consecutive connections in sampled environments. But we should keep in mind that high values of δ also leads to higher rates of false positives.

Chapter 6

Discussions

In this chapter we discuss the results presented in the previous chapter. We try to find out the significance of the results along with the shortcomings. We also discuss possible future work in this field.

6.1 Significance of the Results

Detection based on sufficient data Consecutive connections in stepping stone chains were effectively detected in the different topologies for sFlow packet sampling rates of 1/1 and 1/10. The numbers of detected ON periods and correlated ON periods were sufficient to identify stepping stones.

Content-size based second level check Content-size based correlation reduces the number of false positives. This technique correlates the content sizes of two connections' correlated ON periods. A low correlation score indicates that timing-based detection may have generated a false positive case. This technique is a new proposal and was not part of the previous solutions. This additional step in the detection mechanism reduces the chances of generating false positive alarms.

Effective detection of anomaly due to chaff Edit-distance as well as causality based chaff detection techniques adapt well to the SDN environment when sFlow packet sampling rate is 1/1. For lower packet sampling rate of 1/10, causality based chaff detection becomes ineffective. Edit-distance based chaff detection is, however, effective even in this case.

De-anonymization is a possibility In our experiments, the stepping-stone detection techniques effectively correlate non-consecutive connections

in a chain of stepping stones when sFlow packet sampling rate is 1/1. When sFlow packet sampling rate is 1/10, the timing-based correlation score for non-consecutive connections falls below the threshold value in the tree topology. However, in the clos topology the timing-based technique effectively correlates non-consecutive connections in a chain of stepping stones. The reason for this phenomenon is that, in our experiments, there are fewer switches between two consecutive stepping stones in the tree topology than in the clos topology. For packet sampling rates less than 1/1, the probability of a packet getting sampled at least once increases with the number of switches between the host and the destination. We should keep in mind that clos is a more realistic topology than the tree topology in data-center networks.

Application to non-interactive network traffic As discussed in 3.1.2, attackers can steal video streams by compromising the storage device in a network. Delta-compressed video streams generate a traffic pattern which, although non-interactive in nature, exhibits ON (data) and OFF (no data) periods. Hence, the same detection techniques can be applied to identify the stepping stone, which in this case is the compromised host, by correlating consecutive connections in the chain. This can be used to detect that the video stream is leaking out to the Internet.

6.2 Limitations

Impact of packet sampling on the stepping stone and anomaly detection techniques The timing and content-size based stepping stone detection techniques as well as the edit-distance and causality based chaff detection techniques are affected by sFlow sampling. When the packet sampling rate is set to 1/10, the scores of these detection techniques decrease in all the topologies. In large networks, sFlow packet sampling rates must be much lower than 1/10 to reduce monitoring traffic.

Lack of incorporation of specifics of SDN and NFV environment In our solution, we have not used any property of SDN or NFV which helps in making these detection techniques more effective. Rather, our approach has been naive in trying to find out ways in which existing solutions can be adapted to this new environment. sFlow is a protocol used for monitoring networks in general, and we have used it here alongside OpenFlow. It might be possible to develop new techniques that integrate fully with the SDN and NFV environments.

Limited scalability of the solution We have assumed in our study that a single sFlow collector module will be able to accept all sFlow datagrams sent from the switches in the network and that a single analyzer module will be able to effectively detect all correlations in real-time. In large networks, this might not be the case, and a more scalable solution would be required. Multiple collector and analyzer modules may need to work in parallel on the same forensic data store. Even the data store itself may have to be distributed.

Use of data store leads to lot of disk accesses The forensic data store was implemented using MongoDB, which is accessed by both the collector and analyzer modules. This leads to a lot of disk access. Use of a distributed memory-based caching system like memcached[1] may reduce the latency of disk access.

Continuous bandwidth consumption Our implementation is based on the idea of continuous network monitoring in order to collect forensic data related to the correlation of connections. This results in continuous consumption of network bandwidth as the switches forward sFlow datagrams to the collector. If more of the processing was done at the switches, the bandwidth requirements could be radically reduced.

Lack of real traces and estimate of false positives We simulated stepping stone attacks in different topologies with Mininet. The collected traffic trace was used to analyze the effectiveness of the implemented stepping stone detection techniques. Real attack traces in SDN environments would have increased the accuracy of our results.

The stepping stone detection techniques do not differentiate between attackers and stepping stones in legitimate communication. False positive in our study means two uncorrelated connections reported by the detection mechanisms to be correlated. In our experiments, we did not find a single false positive case. However, if we also considered stepping stones in legitimate communication as false positives, there would be many cases.

Moreover, we found a large number of false negative cases for sFlow sampling rates lower than 1/1. Also, causality based anomaly detection mechanism, unlike edit-distance based anomaly detection mechanism, becomes ineffective at lower sFlow sampling rates and does not mark connections as anomalous even in the presence of chaff in the network traffic.

[1] https://memcached.org/

6.3 Future Work

Scalability The solution can be modified to be more scalable. One can design distributed collector and analyzer modules. In that case, a distributed co-ordination mechanism should be implemented in order to synchronize these applications. Distributed memory cache like memcached can be used to reduce the number of disk accesses by the distributed applications.

Reactive design As we have observed, detection of correlated connection pairs is more efficient at higher packet sampling rates. The detection mechanisms perform poorly when the packet sampling rate is reduced. In order to solve this problem, it is important to identify alternative mechanisms to collect useful information. For example, one may try to incorporate the sFlow polling mechanism or counter values from OpenFlow in the solution. The challenge is that it is easy to collect data at the flow level but difficult to collect data about individual connections within aggregate flows.

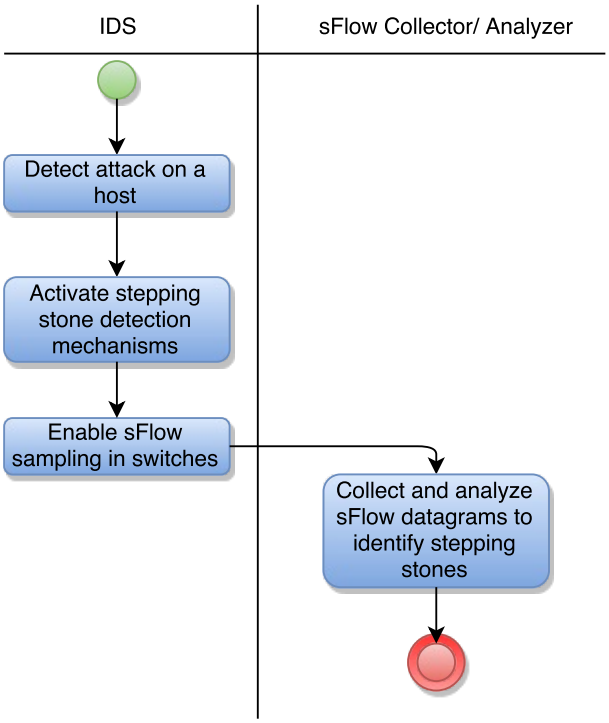

Figure 6.1: IDS triggered reactive stepping stone detection mechanism

An alternative solution might be to make the stepping stone detection mechanism reactive as shown in Figure 6.1. One can use triggered stepping

stone detection instead of continuous monitoring. The deployed IDS, on detection of an attack in an end host, triggers the stepping stone detection mechanisms and enables sFlow forwarding in the switches. For a short period of time, the switches forward all header information to the collector to be analyzed by the analyzer module. Once the chain of stepping stones is identified or the detection mechanism times out, sFlow reporting is disabled in the switches and the detection applications are deactivated. Nevertheless, this scheme might generate a huge amount of redundant traffic in a very short period of time if not efficiently designed. For example, in a clos topology (Figure 4.5) all traffic between hosts passes through the access layer switches or the switches closest to the hosts. Hence, it should be sufficient to enable sFlow in these switches only, thus saving bandwidth. This reduces generated network traffic. Further, an intelligent optimized scheme might be designed to selectively enable sFlow reporting in the switches to assist the stepping stone detection mechanisms.

Botnet detection In our current study, we have only considered detection of stepping stones where the traffic is interactive. Botnets, as already discussed in Chapter 3, can also be used by attacker to launch attacks in a network, and there are various existing techniques to detect bots and botnets. The proposed architecture can be used to analyze the sampled header information, which might reveal potential bot activities and trigger botnet detection mechanisms in SDN and NFV environments.

Chapter 7

Conclusions

In a network of connected hosts, attackers may hop from one compromised host to the other to form a chain of stepping stones before attacking the victim. Stepping stone detection techniques exist to identify the attacker by tracing back through the chain of stepping stones. There are content-matching based [38, 46] active and passive detection techniques for un-encrypted traffic. With the advent of secure shell or SSH, various timing based detection techniques [1, 45, 54, 55] have been proposed. Furthermore, the attackers may try to avoid the detection. For this reason, Crescenzo et al. [11] proposed various anomaly detection techniques to mark connections with deliberately inserted jitter and chaff as anomalous. All of these detection techniques work on interactive traffic where the attacker uses remote terminal sessions to hop from one host to the next.

Stepping stone attacks are still relevant in today's world of automated botnets and malware. In advanced persistent threats or APTs, the attacker spends a lot of time exploring the target network only to steal critical data. In APTs and similar attacks, the attacker often exploits vulnerabilities in a server or host that is within the target network but accessible from outside, compromises it, and uses it as a stepping stone to gain more knowledge o the internals of the network and to launch new attacks. Another typical scenario could be an attacker hijacking live video steams (Figure 3.1) from camera devices and using a compromised host in the target network to relay them to the outside. Here the traffic pattern is also similar to interactive traffic with ON/OFF periods due to the delta compression of the video streams.

SDN and NFV are new avenues of computer networks currently being adopted by the industry. They make the networks programmable and easy to manage. In this study, we investigated how to adapt the stepping stone detection techniques to fit this new model of networks. For network monitoring, we used the sFlow protocol. In our simulations, we used sFlow enabled

Open vSwitches as the forwarding elements. We presented an SDN-based architecture (Figure 4.1) which can be implemented to monitor the data plane for correlated connections and stepping stones. The sFlow enabled switches sample packets passing through them and forward the header information to a central collection and analysis module. The data analysis module removes redundancy in received information and maps sampled headers to connections. It also identifies ON and OFF periods in connections in order to correlate them. The timing based stepping stone detection techniques as well as the anomaly-based chaff detection techniques have been implemented within our analyzer module. Correlations and anomalies are detected in real-time and stored in a forensic data store for future reference.

Our implementation could effectively correlate connections and detect stepping stones for the relatively high packet sampling rates of 1/1 and 1/10. The implemented chaff detection techniques also perform well under similar conditions. Nevertheless, with lower sampling rates, the analyzer misses out critical information and hence the detection becomes ineffective. One solution could be to make the detection mechanism reactive, in which case higher sampling rates for shorter periods of time might be affordable. We also successfully correlated non-consecutive connections in a stepping stone chain. The same technique can be used to correlate the first and last links in an anonymity network to expose an attacker that logs back to its own network via an anonymity service such as Tor.

Future work should be aimed at making the solution scalable. A distributed sFlow controller is needed in large networks. The existing centralized solution should be modified to fit the distributed model. A distributed in-memory cache can be used in place of the forensic data store, which will reduce the number of disk accesses and enhance the performance of the detection mechanisms.

We conclude by claiming that our work can act as a starting point for researchers in the field of network monitoring and stepping stone detection in SDN and NFV environments.

Bibliography

[1] Blum, A., Song, D., and Venkataraman, S. Detection of interactive stepping stones: Algorithms and confidence bounds. In *Recent Advances in Intrusion Detection*, E. Jonsson, A. Valdes, and M. Almgren, Eds. Springer, 2004, pp. 258–277.

[2] Bo, H., Gopalakrishnan, V., Lusheng, J., and Seungjoon, L. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine 53* (2015), 90–97.

[3] Braun, W., and Menth, M. Software-defined networking using OpenFlow: Protocols, applications and architectural design choices. *Future Internet 6* (2014), 302–336.

[4] Chang, W., Wang, A., Mohaisen, A., and Chen, S. Characterizing botnets-as-a-service. In *ACM SIGCOMM Computer Communication Review* (2014), vol. 44, ACM, pp. 585–586.

[5] Chiosi, M., Clarke, D., Willis, P., Reid, A., Feger, J., Bugenhagen, M., Khan, W., Fargano, M., Cui, C., Deng, H., et al. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In *SDN and OpenFlow World Congress* (2012), pp. 22–24.

[6] Chirgwin, R. 'Panama papers' came from email server hack at Mossack Fonseca. `http://www.theregister.co.uk/2016/04/05/email_server_hack_led_to_mossack_fonseca_leak/`. [Online; posted 05-April-2016].

[7] Choi, H., Lee, H., and Kim, H. BotGAD: detecting botnets by capturing group activities in network traffic. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE* (2009), ACM, pp. 2:1–2:8.

[8] Cisco Systems. Introduction to Cisco IOS NetFlow - a technical overview. `http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html`. [Online; accessed 05-February-2016].

[9] Claise, B. Specification of the IP flow information export (IPFIX) protocol for the exchange of IP traffic flow information. RFC 5101, IETF, January 2008.

[10] Daud, A. Y., Ghazali, O., and Omar, M. N. Stepping-stone detection technique for recognizing legitimate and attack connections. In *Proceedings of the 5th International Conference on Computing and Informatics, ICOCI 2015* (2015), pp. 440–446.

[11] Di Crescenzo, G., Ghosh, A., Kampasi, A., Talpade, R., and Zhang, Y. Detecting anomalies in active insider stepping stone attacks. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA) 2*, 1 (2011), 103–120.

[12] Dietrich, C. J., Rossow, C., Freiling, F. C., Bos, H., Van Steen, M., and Pohlmann, N. On botnets that use DNS for command and control. In *2011 Seventh European Conference on Computer Network Defense* (2011), IEEE, pp. 9–16.

[13] Doria, A., Hadi Salim, J., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., and Halpern, J. Forwarding and control element separation (ForCES) protocol specification. RFC 5810, IETF, March 2010.

[14] Ehrlich, W. K., Karasaridis, A., Hoeflin, D. A., and Liu, D. Detection of spam hosts and spam bots using network flow traffic modeling. In *3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats* (2010).

[15] Gilbert, J. I. Scalable wavelet-based active network stepping stone detection. DTIC document, 2012.

[16] Gu, G., Perdisci, R., Zhang, J., Lee, W., et al. BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium (Security'08)* (2008), pp. 139–154.

[17] Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., and Lee, W. BotHunter: Detecting malware infection through IDS-driven

dialog correlation. In *Proceedings of the 16th USENIX Security Symposium (Security'07)* (2007), pp. 167–182.

[18] Gu, G., Zhang, J., and Lee, W. Botsniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium, NDSS* (2008).

[19] Hakiri, A., and Berthou, P. Leveraging SDN for the 5G networks: Trends, prospects and challenges. In *Software Defined Mobile Networks (SDMN) - Beyond LTE Network Architecture*, M. Liyanage, A. Gurtov, and M. Ylianttila, Eds. WILEY, 2015, ch. 5, pp. 61–80.

[20] He, T., and Tong, L. Detecting encrypted stepping-stone connections. In *IEEE Transactions on Signal Processing - Volume 55, Issue 5* (2007), IEEE, pp. 1612 – 1623.

[21] Jacobson, V. Congestion avoidance and control. In *ACM SIGCOMM Computer Communication Review* (1988), vol. 18, ACM, pp. 314–329.

[22] Kalt, C. Internet relay chat: Client protocol. RFC 2812, IETF, April 2000.

[23] Karasaridis, A., Rexroad, B., and Hoeflin, D. A. Wide-scale botnet detection and characterization. *USENIX HotBots 7* (2007).

[24] Kirkpatrick, K. Software-defined networking. *Commun. ACM 56*, 9 (Sept. 2013), 16–19.

[25] Lakshman, T. V., Nandagopal, T., Ramjee, R., Sabnani, K., and Woo, T. The SoftRouter architecture. In *ACM SIGCOMM Workshop on Hot Topics in Networking* (2004).

[26] Maymounkov, P., and Mazieres, D. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[27] Members of the 5G Infrastructure Association. 5G vision. Tech. rep., 5G Infrastructure Association, February 2015. Supported by the European Commission.

[28] Nagaraja, S., Mittal, P., Hong, C.-Y., Caesar, M., and Borisov, N. BotGrep: Finding P2P bots with structured graph analysis. In *USENIX Security Symposium* (2010), pp. 95–110.

[29] Nechaev, B., and Gurtov, A. Internet botnets: A survey of detection techniques. In *Case Studies in Secure Computing - Achievements and Trends*, B. Issac and N. Israr, Eds. Auerbach Publications, 2014, ch. 20, pp. 405–424.

[30] Open Networking Foundation. OpenFlow. `https://www.opennetworking.org/sdn-resources/openflow`. [Online; accessed 04-February-2016].

[31] Open Networking Foundation. SDN architecture overview. Tech. Rep. ONF TR-504, Open Networking Foundation, November 2014.

[32] Paxson, V., and Floyd, S. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking (ToN) 3*, 3 (1995), 226–244.

[33] Peng, P., Ning, P., Reeves, D. S., and Wang, X. Active timing-based correlation of perturbed traffic flows with chaff packets. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on* (2005), IEEE, pp. 107–113.

[34] Phaal, P., Panchen, S., and McKee, N. InMon Corporation's sFlow: A method for monitoring traffic in switched and routed networks. RFC 3176, IETF, September 2001.

[35] Pyun, Y. J., Park, Y. H., Wang, X., Reeves, D. S., and Ning, P. Tracing traffic through intermediate hosts that repacketize flows. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications* (2007), pp. 634–642.

[36] Risen, T. Mossack Fonseca blames Panama papers leak on hackers. `http://www.usnews.com/news/articles/2016-04-06/mossack-fonseca-blames-panama-papers-leak-on-hackers`. [Online; posted 06-April-2016].

[37] Roesch, M., et al. Snort: Lightweight intrusion detection for networks. In *13th Systems Administration Conference (LISA'99)* (1999), vol. 99, USENIX, pp. 229–238.

[38] Staniford-Chen, S., and Heberlein, L. T. Holding intruders accountable on the internet. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 1995), IEEE Computer Society, pp. 39–49.

[39] Strayer, T. W., Jones, C., Schwartz, B., Edwards, S., Milliken, W., and Jackson, A. Efficient multi-dimensional flow correlation. In *32nd IEEE Conference on Local Computer Networks (LCN'07)* (2007), IEEE, pp. 531–538.

[40] Temperton, J., and Burgess, M. The security flaws at the heart of the Panama papers. `http://www.wired.co.uk/news/archive/2016-04/06/panama-papers-mossack-fonseca-website-security-problems`. [Online; posted 06-April-2016].

[41] The Internet Defense League. Tor project: Anonymity online. `https://www.torproject.org/`. [Online; accessed 03-February-2016].

[42] Villamarín-Salomón, R., and Brustoloni, J. C. Bayesian bot detection based on DNS traffic similarity. In *Proceedings of the 2009 ACM symposium on Applied Computing* (2009), ACM, pp. 2035–2041.

[43] Wang, X., Chen, S., and Jajodia, S. Tracking anonymous peer-to-peer VoIP calls on the internet. In *Proceedings of the 12th ACM conference on Computer and communications security* (2005), ACM, pp. 81–91.

[44] Wang, X., Reeves, D. S., Ning, P., and Feng, F. Robust network-based attack attribution through probabilistic watermarking of packet flows. Tech. rep., Technical Report TR-2005-10, Department of Computer Science, NC State Univ, 2005.

[45] Wang, X., Reeves, D. S., and Wu, S. F. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In *Computer Security - ESORICS 2002*, D. Gollmann, G. Karjoth, and M. Waidner, Eds. Springer, 2002, pp. 244–263.

[46] Wang, X., Reeves, D. S., Wu, S. F., and Yuill, J. Sleepy watermark tracing: An active network-based intrusion response framework. In *Trusted Information*. Springer, 2002, pp. 369–384.

[47] Wijesinghe, U., Tupakula, U., and Varadharajan, V. Botnet detection using software defined networking. In *Telecommunications (ICT), 2015 22nd International Conference on* (2015), IEEE, pp. 219–224.

[48] Wu, H.-C., and Huang, S.-H. S. Performance of neural networks in stepping-stone intrusion detection. In *Networking, Sensing and Control,*

2008. ICNSC 2008. IEEE International Conference on (2008), IEEE, pp. 608–613.

[49] Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., and Osipkov, I. Spamming botnets: signatures and characteristics. In *ACM SIGCOMM Computer Communication Review* (2008), vol. 38, ACM, pp. 171–182.

[50] Yadav, S., Reddy, A. K. K., Reddy, A., and Ranjan, S. Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement* (2010), ACM, pp. 48–61.

[51] Yang, J., and Huang, S.-H. S. Correlating temporal thumbprint for tracing intruders. In *Proceedings of 3rd International Conference on Computing, Communications and Control Technologies* (2005), pp. 236–241.

[52] Yang, J., and Huang, S.-H. S. Matching TCP/IP packets to detect stepping-stone intrusion. *International Journal of Computer Science and Network Security 6*, 4 (2006), 269–276.

[53] Ylonen, T. The secure shell (SSH) transport layer protocol. RFC 4253, IETF, January 2006.

[54] Yung, K. H. Detecting long connection chains of interactive terminal sessions. In *Recent Advances in Intrusion Detection*, A. Wespi, G. Vigna, and L. Deri, Eds. Springer, 2002.

[55] Zhang, Y., and Paxson, V. Detecting stepping stones. In *Proceedings of the 9th Conference on USENIX Security Symposium - Volume 9* (Berkeley, CA, USA, 2000), SSYM'00, USENIX Association.