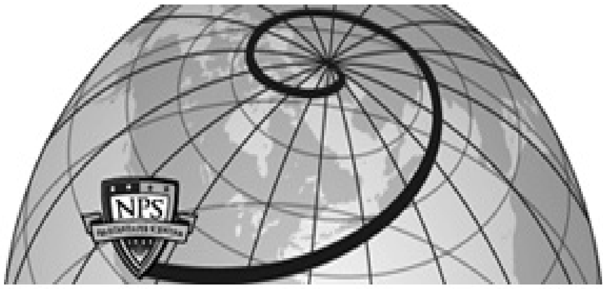


2022-12

ANALYSIS OF CLIENT-SIDE ATTACKS THROUGH DRIVE-BY HONEYPOTS

Foley, Brian A.

Monterey, CA; Naval Postgraduate School

https://hdl.handle.net/10945/71460

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ANALYSIS OF CLIENT-SIDE ATTACKS THROUGH DRIVE-BY HONEYPOTS

by

Brian A. Foley

December 2022

Thesis Advisor:	Neil C. Rowe
Co-Advisor:	Thuy D. Nguyen

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE ANALYSIS OF CLIENT-SIDE ATTACKS THROUGH DRIVE-BY HONEYPOTS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Brian A. Foley			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Client-side cyberattacks on Web browsers are becoming more common relative to server-side cyberattacks. This work tested the ability of the honeypot (decoy) client software Thug to detect malicious or compromised servers that secretly download malicious files to clients, and to classify what it downloaded. Prior to using Thug we did TCP/IP fingerprinting to assess Thug's ability to impersonate different Web browsers, and we created our own malicious Web server with some drive-by exploits to verify Thug's functions; Thug correctly identified 85 out of 86 exploits from this server. We then tested Thug's analysis of delivered exploits from two sets of real Web servers; one set was obtained from random Internet addresses of Web servers, and the other came from a commercial blacklist. The rates of malicious activity on 37,415 random websites and 83,667 blacklisted websites were 5.6% and 1.15%, respectively. Thug's interaction with the blacklisted Web servers found 163 unique malware files. We demonstrated the usefulness and efficiency of client-side honeypots in analyzing harmful data presented by malicious websites. These honeypots can help government and industry defenders to proactively identify suspicious Web servers and protect users.

14. SUBJECT TERMS honeypot, honeyclient, client-side, drive-by, Thug, cyberattacks, malware			**15. NUMBER OF PAGES** 71
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

ANALYSIS OF CLIENT-SIDE ATTACKS THROUGH DRIVE-BY HONEYPOTS

Brian A. Foley
Lieutenant, United States Navy
BA, Rutgers, State University of New Jersey, 2016

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2022

Approved by: Neil C. Rowe
Advisor

Thuy D. Nguyen
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Client-side cyberattacks on Web browsers are becoming more common relative to server-side cyberattacks. This work tested the ability of the honeypot (decoy) client software Thug to detect malicious or compromised servers that secretly download malicious files to clients, and to classify what it downloaded. Prior to using Thug we did TCP/IP fingerprinting to assess Thug's ability to impersonate different Web browsers, and we created our own malicious Web server with some drive-by exploits to verify Thug's functions; Thug correctly identified 85 out of 86 exploits from this server. We then tested Thug's analysis of delivered exploits from two sets of real Web servers; one set was obtained from random Internet addresses of Web servers, and the other came from a commercial blacklist. The rates of malicious activity on 37,415 random websites and 83,667 blacklisted websites were 5.6% and 1.15%, respectively. Thug's interaction with the blacklisted Web servers found 163 unique malware files. We demonstrated the usefulness and efficiency of client-side honeypots in analyzing harmful data presented by malicious websites. These honeypots can help government and industry defenders to proactively identify suspicious Web servers and protect users.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

API	applications programming interface
CPU	central processing unit
CVE	common vulnerabilities and exposures
DDoS	distributed denial of service
DNS	domain name system
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDS	intrusion-detection system
IP	Internet Protocol
JSON	JavaScript object notation
NoSQL	non-SQL
SQL	Structured Query Language
SSH	secure shell
TCP	Transport Control Protocol
TCP/IP	Internet Protocol Suite
TTL	time-to-live
URI	uniform resource identifier
URL	uniform resource locator
VM	virtual machine
VNC	virtual network computing

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisors, Dr. Neil Rowe and Professor Thuy Nguyen. They provided me with guidance and direction from beginning to end. Their efforts fueled my enthusiasm, challenged my critical thinking, and encouraged my creativity. It was my great pleasure to be mentored by them and accomplish this work.

Next, I would like to thank my friends in the 212 CSO cohort at NPS. They kept me motivated to continue working, assisted when it was needed, and shared in the struggles of the bad times while celebrating the good ones. A special thank you to Kelly Rosnick. You know what you did.

Lastly, I would like to thank my wonderful wife, Erin. She supported me throughout my entire graduate school experience with her unconditional love and care.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The Internet serves as essential global communication, used by individuals, companies, governments, and other organizations. These communications serve essential purposes, such as sharing news and information, allowing businesses to function, operating critical infrastructure, and supporting the global economy. However, the interconnectedness of the Internet exposes billions of users as targets to cyberattacks. Competition between cybersecurity experts and malicious actors has occurred since the Internet's start. New software and technologies are constantly scrutinized by malicious actors for vulnerabilities to exploit. New countermeasures developed to combat exploited vulnerabilities are also subject to scrutiny by malicious actors to identify new vulnerabilities and exploits, creating an endless cycle. A recent report by SonicWall shows an 11% increase in malware attacks in 2022, totaling 2.8 billion attacks (Conner, 2022). Recently, drive-by download attacks have increased, affecting thousands of users who were unaware that malicious code was downloaded to their machines while they were browsing the Web (Lake, 2019). These are attacks on visitors to Web sites.

In 2016, several high-profile Web publishers were affected through compromised advertising servers, causing tens of thousands of users to be infected with backdoor trojans and ransomware by malicious actors (Lake, 2019). The websites affected included The New York Times, MSN, the BBC, the NFL, and Comcast's Xfinity Web sites. All the sites were using legitimate advertising servers that malicious actors compromised. In the attack, hackers delivered malicious advertisements to unsuspecting users, which caused the execution of "drive-by" downloads and installation of backdoor Bedep Trojan and TeslaCrypt ransomware. In July 2019, security researchers discovered malicious actors used drive-by downloads to deliver the Eris ransomware to unsuspecting users by exploiting a browser Shockwave vulnerability (Lake, 2019). In this attack, malicious actors compromised the PopCash advertisement network and redirected users to the RIG exploitation kit on their malicious Web server. Their RIG kit exploited a browser Shockwave vulnerability; if successful, the kit would download and install the Eris ransomware, which encrypted the user's files and demanded payment for the decryption key.

These attacks are a few examples of many; therefore, developing new software and technologies is essential to combat the rising cyber-attack threats. Honeypots (decoy information systems) are an essential tool for cybersecurity experts to examine the behaviors and methods used by malicious actors and develop new software, technology, and methods to counter their efforts.

A. PROBLEM STATEMENT

The number of client-side attacks has increased recently, with malicious actors targeting vulnerable clients rather than servers using drive-by downloads and other vectors to compromise clients. We will test whether the client-side honeypot Thug can successfully deceive malicious or compromised Web servers in collecting drive-by vectors and identifying relevant or common threats.

B. MOTIVATION AND BENEFITS OF STUDY

This research aimed to identify drive-by download exploits and other attack methods against Web users. This study will inform cybersecurity experts in government and industry of the current extent of client-side attack risk for users browsing the Internet. This study will also benefit future use of client-side honeypot software by assessing its effectiveness in recognizing exploits and their ability to deceive malicious or compromised servers. Assessing effectiveness will also identify areas of improvement for client-side honeypots.

C. ORGANIZATION OF THE THESIS

Chapter II summarizes honeypots, client-side attacks, and work related to our research. Chapter III describes Thug, our client-side honeypot, and supporting software used for our experiments. Chapter IV discusses our experiments and Chapter V shows the analysis of our collected data. Chapter VI provides the conclusions from our work and ideas for future research.

II. BACKGROUND

Honeypots are cybersecurity measures that act as decoys for detecting and analyzing intrusion behaviors. They help analyze behaviors of attackers, such as ingress vectors and identify code used for compromising systems. Honeypots differ based on varying levels of interaction and ways to configure and apply them. This chapter covers honeypot software and techniques, other software used in this thesis project, and previous work in this field.

A. OVERVIEW ON HONEYPOTS

A honeypot distracts or lures attackers through deception. Honeypots can monitor actions such as keystrokes, authentication attempts, files accessed or modified, and executed processes. The value of the honeypot is in how much it can be attacked, probed, or compromised (Joshi & Sardana, 2011). Honeypots emulate different operating systems and services to lure or entrap attackers. Attackers may believe they are compromising a legitimate system, but they will not get very far, and the honeypot logs the methods the attacker uses. Honeypots have advantages over traditional intrusion-detection systems (IDS) and firewalls in that they collect richer data because any access to a honeypot besides an administrative connection is an intrusion, and they can also identify new types of threats and anomalies.

Honeypot implementations can be categorized by interaction level, environment in which they operate, equipment deployment type, and the role the honeypot fulfills. Low-interaction honeypots provide minimal interaction with fake services and are safer than other types of honeypots (Rowe & Rrushi, 2016). Low-level honeypots can identify scans and other automated attacks, deceive simple attacks, distract attackers from more essential systems, and collect attack signatures and behaviors.

Medium-interaction honeypots are more sophisticated and require more knowledge and expertise to configure and run. These honeypots help receive and emulate responses to payloads from attackers. A medium-interaction honeypot behaves more like a legitimate system to entice attackers to stay longer, enabling more intelligence collection on attack attempts.

High-interaction honeypots typically offer genuine services to lure attackers, thus carrying a higher risk to the attack target. They can provide complete system access and collect full details on attacker behavior and exploits. These honeypots are challenging to deploy due to the tools required to configure them and the necessary safeguards that must be implemented to keep the system isolated from the legitimate network. However, they can provide valuable information on new techniques and exploits not seen in traditionally protected networks.

Honeypots can also be distinguished by their intended environment, usually either for protection or research. Commercial companies use honeypots to protect their organization and networks. Research honeypots alert cybersecurity experts of vulnerabilities or methods attackers use to access their systems. Honeypots can also be distinguished by their physical or virtual form. Physical honeypots are machines running a real operating system connected to a network through a single IP address; they are suitable for high interaction due to their legitimate operating system and hardware, but they have a high cost and require an infrastructure to configure and maintain. Virtual honeypots only simulate a machine, and are becoming more common as virtualization allows easier deployment of multiple honeypots and enables honeypots to cover a bigger address space.

Lastly, honeypots can be distinguished based on their roles. Server-side honeypots are passive, allowing attackers to connect to them, but do not start traffic unless compromised (Joshi & Sardana, 2011). Client-side honeypots are active, starting connections and collecting exploits targeted at client applications, often Web browsers, by hostile servers (Qassrawi & Zhang, 2011). Thug, the product we studied in this thesis, is a client-side honeypot that starts connections to identify exploits of vulnerabilities in different browser implementations (Dell'Aera, 2022).

The Honeynet Project is an international nonprofit security research organization investigating attacks and exploits and providing open-source tools to improve cybersecurity (honeynet.org, 2022). Their website links include several kinds of honeypot software, including server and client-side honeypots and tools for malware analysis. They developed Thug.

B. WEB CRAWLERS

Web crawling is essential for client-side honeypots, which need to search websites to find malicious Web links (URLs). Web crawlers search and index content found on the Internet, and are widely used by commercial Web indexers like Bing and Google. Web crawlers work by discovering new URLs, examining and categorizing their contents, and finding hyperlinks to other Web pages to which to crawl next.

Usually, Web crawlers work from a list of Web links provided by commercial Web indexers to analyze (Ikinci et al., 2008). However, this method means popular Web pages are selected at higher rates, creating bias in the sampling. Other Web crawling methods include keyword searches to select "seeds" for crawling or generating random IP addresses (Invernizzi et al., 2012).

C. CLIENT-SIDE ATTACKS

Client-side Web attacks ("drive-by exploits") occur when Web users visit a webpage that delivers an HTML document containing malicious code. The malicious code can exploit vulnerabilities in the Web browser, browser plugins, or operating system to compromise the user's Web browser. Then malicious actors can download and execute additional malware, compromising the user's system. This process can occur without the user's knowledge by simply visiting a Web page.

A drive-by exploit is a four-stage process shown in Figure 1 (Le et al., 2013). Attackers first load malicious code into the HTML documents of a website. Attackers can lure visitors in several ways: by sending spam email with links to their servers, abusing search engines to report their pages through search engine optimization, using social media to publish their links, or modifying legitimate Web servers. Users then visit the malicious or compromised servers and unknowingly retrieve the attacker's malware. Usually, attackers target a specific browser or operating-system vulnerability for exploitation. Therefore they often try to detect the user's system, version, and installed software to decide which version of the malicious content to deliver. Software often exploited includes Adobe Acrobat, Adobe Flash Player, Apple QuickTime, Java, Microsoft ActiveX controls, and Microsoft Silverlight.

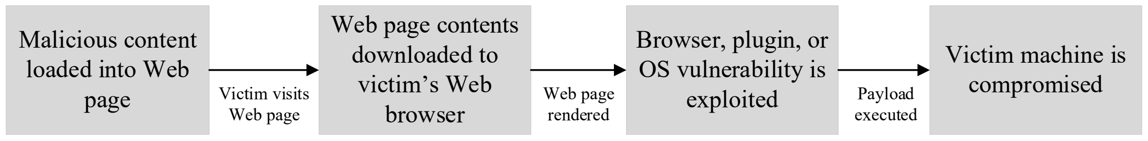

Malicious content loaded into Web page
Victim visits Web page
Web page contents downloaded to victim's Web browser
Web page rendered
Browser, plugin, or OS vulnerability is exploited
Payload executed
Victim machine is compromised

Figure 1. Four-stage process of a drive-by download. Adapted from Le et al. (2013).

D. MALWARE

Malware is code that has been added, removed, or changed by a malicious actor to intentionally harm an information system or alter its intended function. Many variations and types of malware occur, each with particular goals. Many of these goals are financially oriented, from collecting information on unsuspecting users and selling that information to advertisers, stealing banking information to permit fraud, or encrypting a user's data and holding it hostage for ransom. Other malicious actors create malware to cause havoc and destruction without financial gain. One can broadly classify malware into several categories (Caviglione et al., 2021; Namanya et al., 2018).

- A *virus* is a self-replicating malicious program that copies itself by spreading to and infecting other systems. Viruses are passive and need action from the user to replicate, such as through sharing infected media or email. Viruses can produce a range of malicious actions, like collecting information from a user for advertising purposes, harming the underlying system, or connecting the infected computer to a botnet.
- *Worms* are like viruses but do not require human interaction to spread. Instead, they scan open ports and software vulnerabilities to find exploits that allow propagation.

- A *Trojan* is a program that appears as legitimate software to a user but has malicious functions. They can install other malware, steal information, or allow attackers to access the victim's machine remotely, depending on the payload. Like viruses, Trojans rely on interaction with the victim for activation and spread.
- *Spyware* are malicious programs that abuse the infected operating system's functions to spy on the victim's actions, by logging keystrokes, recording user behavior, or logging the user's Web activities. This information is sent to the malicious actor, who can use the information to access bank accounts or serve directed advertisements.
- *Adware* are programs that deliver advertisements to the user, usually pop-ups or dialog boxes. While adware may not have malicious intentions, they can annoy users.
- *Rootkits* are complex programs that hide in an infected computer's privileged processes, letting them take complete control of the infected computer. Because of their privileges, they are very invasive and difficult to remove. Rootkits can accomplish many malicious actions, notably controlling the victim system and adding it to a botnet.
- *Bots* are programs configured to perform specific actions; however, they are used by malicious actors to accomplish malevolent goals, such as forming botnets. Malicious botnets are often used for distributed denial of service (DDoS) attacks.
- *Ransomware* are malicious programs that hold the file system of an infected computer hostage while demanding ransom for the release of the files. This is usually done with strong encryption techniques, where the program encrypts files, leaving the victim unable to access any of their critical data. These programs are used by malicious actors for extortion, often in the form of cryptocurrency.

- A relatively new malware threat is *cryptojacking*, which uses unauthorized access to the victim's machine to mine cryptocurrencies and generate revenue for the malicious actor. While the victim may be unaware of its presence, these programs burden the host machine by increasing its power demands and causing unnecessary wear.

All these malware types can be installed and executed through drive-by download exploits.

E. PREVIOUS WORK

1. Detecting Malicious Web Servers with Honeypot Clients Using Keyword Searches

A related project compared low-interaction and high-interaction honeypot clients (Qassrawi and Xhang, 2011). It used HoneyC (Seifert, 2006) as its low-interaction honeypot and Capture-HPC (Seifert & Steenson, 2006) as its high-interaction honeypot. The researchers used the commercial tool SiteAdvisor to check their results by analyzing the same selected Web sites. However, SiteAdvisor does not operate as a honeypot client, but downloads portions of Web pages and scans them for signatures of known malicious software. Ten keywords were used for URL selection for testing, identifying about 20,000 websites. These were separated into seven categories shown in Figure 2.

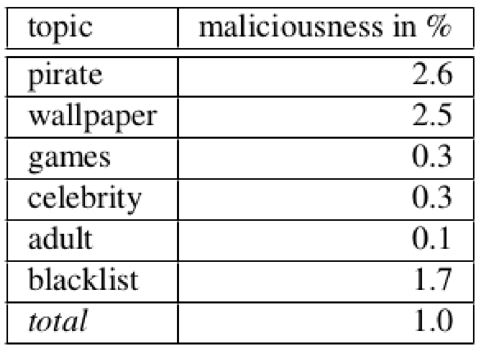
Category	**Malicious URL Rate %**		
	SiteAdvisor	HoneyC	Capture
Warez	9.3	3.9	4.3
Cracks	1.9	1.1	1.3
Screen Saver	7.3	3.1	3.9
Adult	0.7	0.2	0.3
Games	1.0	0.6	0.8
Celebrity	5.1	0.5	1.2
Wallpaper	6.67	3.5	3.8

Figure 2. Client honeypots and SiteAdvisor malicious identification results from 20,000 analyzed websites. Source: Qassrawi and Zhang (2011).

The results show a significant difference between malicious URLs identified by SiteAdvisor compared to the two honeypots. However, SiteAdvisor uses additional information beyond website visits to determine maliciousness: spam and phishing site reports, request site analyses, and Domain Name System (DNS) information. The researchers concluded that combining the technologies of high-interaction client-side honeypots with low-interaction ones is necessary. They suggested using low-interaction honeypots to quickly identify suspicious behavior to pass on to the high-interaction honeypot for further analysis.

2. Detecting Malicious Web Servers with Honeypot Clients Using Web Search Seeding

A honeypot client Monkey-Spider analyzed crawled webpages through search-engine seeding (Ikinci et al., 2008). Monkey-Spider is a low-interaction honeypot client that separates Web crawling from webpage analysis for malicious code using ClamAV (Ikinci, 2008). For starting sites for crawling, they used the Web Services API (applications programming interface) for Google, Yahoo, and MSN search and collected the first 1,000 site results from five search keywords as well as URLs from commercial blacklists (lists of known malicious sites). These sites were their initial Web crawler visits ("seeds"). The researchers used 20,457 seeds downloaded from 20,005,756 URLs during the Web crawling. The fractions of malware they found out of the total files retrieved from websites are given in Figure 3.



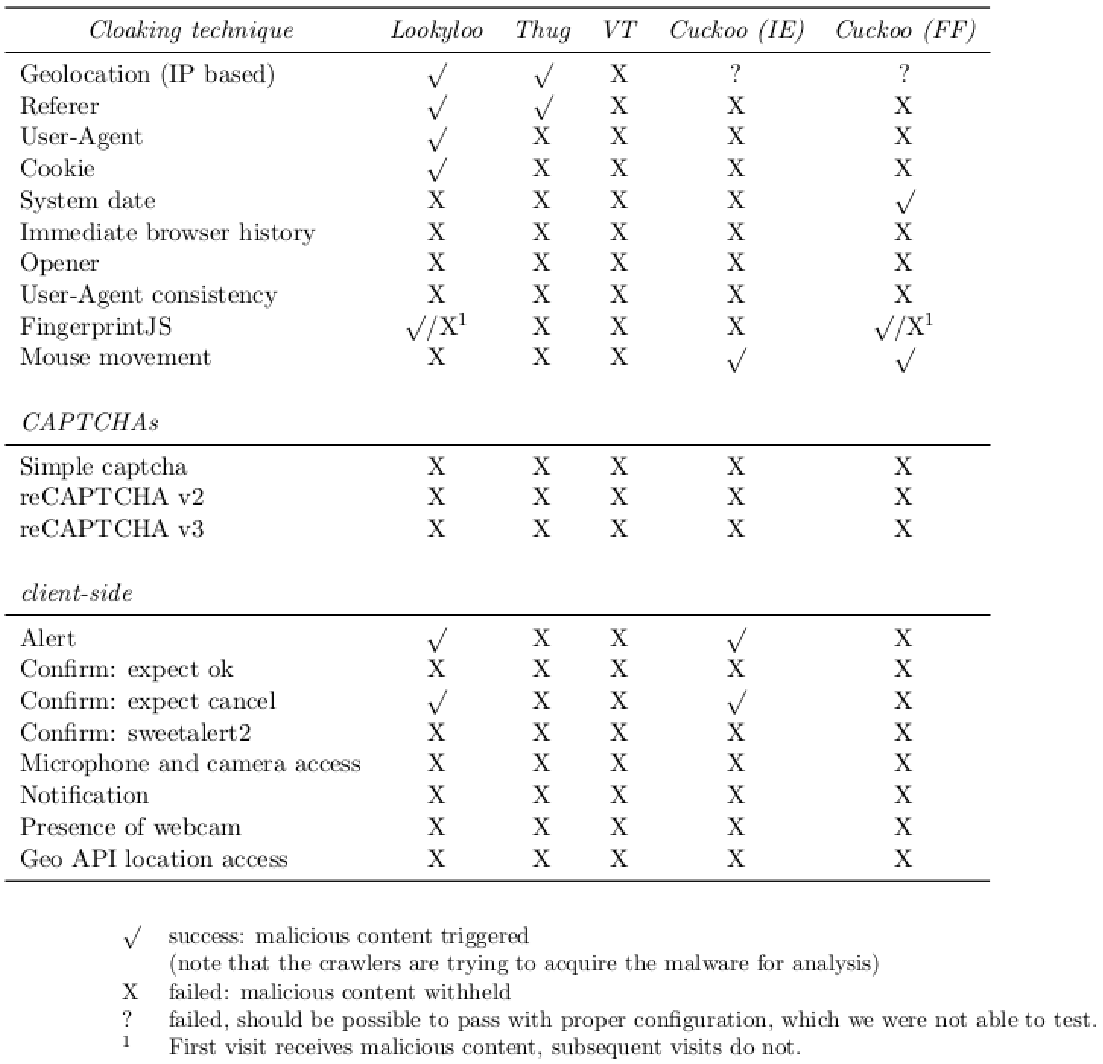
topic	maliciousness in %
pirate	2.6
wallpaper	2.5
games	0.3
celebrity	0.3
adult	0.1
blacklist	1.7
total	1.0

Figure 3. Monkey-Spider Web crawling results. Source Ikinci et al. (2008).

These results show fewer malicious websites compared to a 2007 report from SiteAdvisor, which identified 4.1 percent of websites as malicious (Keats et al., 2007). This discrepancy could be due to the Web crawler focusing on extracting URLs over downloading their content, which meant fewer malicious files downloaded. Another issue they saw was duplicate visits to popular websites like YouTube and Amazon. Since these sites are popular, they less often have malware. The malware collected from their experiment consisted mainly of Trojans and adware, the severity of which they did not report. These researchers concluded that their methods could identify malicious websites faster than high-interaction honeypot clients.

3. Cloaking Malicious Web Servers to Avoid Honeypot Client Detection

Another project studied the effectiveness of client-side honeypots in identifying malicious webservers that used fingerprinting and bot-detection techniques (Pinoy et al., 2021). They used two honeypot clients and two kinds of analysis software. One client was the low-interaction client Thug, and the other was the high-interaction honeypot client Cuckoo Sandbox (Stichting Cuckoo Foundation, 2019). The tools used were Lookyloo, which captures webpages and redirection data (Lookyloo.eu, 2022), and VirusTotal, a common cybersecurity tool for identifying malicious code (VirusTotal.com, 2022). The project created a custom Web server with the Django Python Web framework using 21 cloaking methods to prevent analysis by clients. All four tools had difficulty pulling malicious content from the Web server due to its cloaking techniques, as shown in Figure 4. Also, most tools did not support newer technologies, applications programming interfaces, and browser versions. This suggests modern websites can effectively cloak their malicious-download techniques from honeypot clients and other analysis tools.



Cloaking technique	*Lookyloo*	*Thug*	*VT*	*Cuckoo (IE)*	*Cuckoo (FF)*
Geolocation (IP based)	√	√	X	?	?
Referer	√	√	X	X	X
User-Agent	√	X	X	X	X
Cookie	√	X	X	X	X
System date	X	X	X	X	√
Immediate browser history	X	X	X	X	X
Opener	X	X	X	X	X
User-Agent consistency	X	X	X	X	X
FingerprintJS	√/X[1]	X	X	X	√/X[1]
Mouse movement	X	X	X	√	√
CAPTCHAs					
Simple captcha	X	X	X	X	X
reCAPTCHA v2	X	X	X	X	X
reCAPTCHA v3	X	X	X	X	X
client-side					
Alert	√	X	X	√	X
Confirm: expect ok	X	X	X	X	X
Confirm: expect cancel	√	X	X	√	X
Confirm: sweetalert2	X	X	X	X	X
Microphone and camera access	X	X	X	X	X
Notification	X	X	X	X	X
Presence of webcam	X	X	X	X	X
Geo API location access	X	X	X	X	X

√	success: malicious content triggered (note that the crawlers are trying to acquire the malware for analysis)
X	failed: malicious content withheld
?	failed, should be possible to pass with proper configuration, which we were not able to test.
1	First visit receives malicious content, subsequent visits do not.

Figure 4. Web server cloaking techniques detected by client honeypots. Source: Pinoy et al. (2021)

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY AND DESIGN

This chapter discusses honeypot software and technologies relevant to our research and methods for analyzing Web servers for drive-by download exploits.

A. THUG

Thug is a Python-based low-interaction honeypot client that does static and dynamic analyses to inspect suspicious malware (Dell'Aera, 2022). Thug uses the Google V8 JavaScript engine (Google.com, 2022) wrapped through STPyV8 (Dell'Aera & Syme, 2019/2022) to analyze malicious JavaScript code, and uses Libemu (Dell'Aera et al., 2022) wrapped through Pylibemu (Dell'Aera et al., 2022) to detect and emulate shell codes. Thug currently emulates 42 different browser types and provides 90 vulnerability plugins for analysis, as in Table 1. Browser type is passed by a command-line argument to Thug when running, with additional vulnerability plugins and the target URL for analysis. After analyzing the specified URLs and collecting data from the Web site, Thug stores its results in a NoSQL MongoDB database (Mongo.org, 2022) running on either the host machine or a separate system. Thug's operation is summarized in Figure 5. We chose this client-side honeypot based on its capabilities, open-source license, and active maintenance and support.

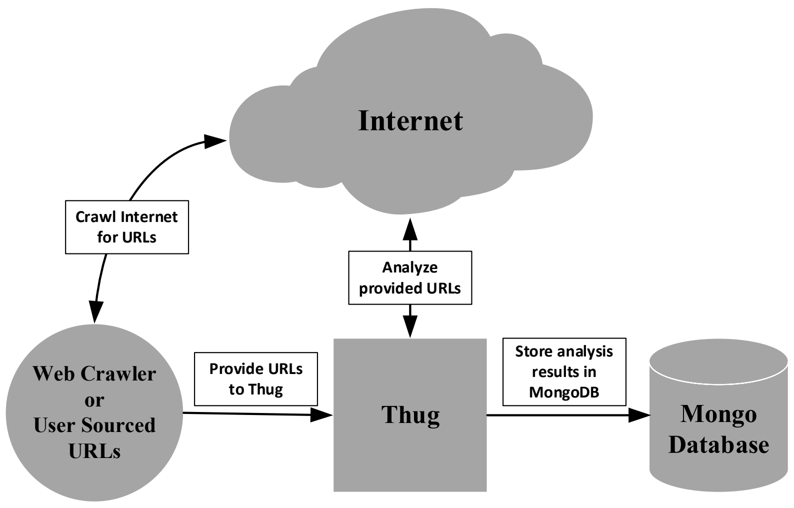


Figure 5. Thug operation diagram

Table 1. Browser types emulated by Thug. Adapted from Dell'Aera (2022).

Web Browser	Operating System	Broswer Version
Internet Explorer	Windows XP	Internet Explorer 6.0
		Internet Explorer 6.1
		Internet Explorer 7.0
		Internet Explorer 8.0
	Windows 2000	Internet Explorer 6.0
		Internet Explorer 8.0
	Windows 7	Internet Explorer 8.0
		Internet Explorer 9.0
		Internet Explorer 10.0
Chrome	Windows XP	Chrome 20.0.1132.47
	Windows 7	Chrome 20.0.1132.47
		Chrome 40.0.3987.116
		Chrome 45.0.2454.85
		Chrome 49.0.2623.87
	MacOS X 10.7.4	Chrome 19.0.1084.54
	MacOS X 10.15.3	Chrome 80.0.3977.116
	MaxOS X 10.17.7	Chrome 97.0.4692.99
	Linux	Chrome 26.0.1410.19
		Chrome 30.0.1599.15
		Chrome 44.0.2403.89
		Chrome 54.0.2840.100
		Chrome 98.0.4758.102
	Samsung Galaxy S II, Android 4.0.3	Chrome 18.0.1025.166
		Chrome 25.0.1364.123
	Samsung Galaxy S II, Android 4.1.2	Chrome 29.0.1547.59
	Google Nexus, Android 4.0.4	Chrome 18.0.1025.133
	iPad, iOS 7.1	Chrome 33.0.1750.21
	iPad, iOS 7.1.1	Chrome 35.0.1916.41
	iPad, iOS 7.1.2	Chrome 37.0.2062.52
	iPad, iOS 8.0.2	Chrome 38.0.2125.59
	iPad, iOS 8.1.1	Chrome 39.0.2171.45
	iPad, iOS 8.4.1	Chrome 45.0.2454.68
	iPad, iOS 9.0.2	Chrome 46.0.2490.73
	iPad, iOS 9.1	Chrome 47.0.2526.70
Firefox	Windows XP	Firefox 12.0
	Windows 7	Firefox 3.6.13
	Linux	Firefox 19.0
		Firefox 40.0
Safari	Windows XP	Safari 5.1.7
	Windows 7	
	MacOS X 10.7.2	Safari 5.1.1
	MacOS X 11.2.3	Safari 14.0.3
	iPad, iOS 7.0.4	Safari 7.0
	iPad, iOS 8.0.2	Safari 8.0
	iPad, iOS 9.1	Safari 9.0

B. SUPPORTING TOOLS

MongoDB is an open-source document-oriented NoSQL database. It records data with field-name and value pairs similar to JSON objects, and a document can include other documents, arrays, or arrays of documents. Thug uses MongoDB to store data from its interactions with websites. It records URLs analyzed, analysis type, connections made, behaviors observed, codes identified, cookies downloaded, connection graphs, locations, and certificates. The collection schema is shown in Table 2.

Table 2. MongoDB Thug collection schema. Adapted from Dell'Aera (2022).

MongoDB collection	Data type stored
Analyses	Logs data about the analysis type, including Thug version, personality, and plugins enabled
Behaviors	Logs behavior of the Web page
Certificates	Logs certificates collected
Codes	Logs extracted code from the analysis
Connections	Logs redirections during analysis
Cookies	Logs cookies collected
Exploits	Logs identified exploits
Graphs	Logs the analysis JSON exploit graph
Locations	Logs the content stored at each URL
URLs	Logs URLs analyzed
fs.files	Stores metadata of collected file samples
fs.chunks	Stores file chunks of collected file samples

Thug stores collected file samples as GridFS chunks in MongoDB, in Figure 6. The top diagram shows how Thug uses MongoDB to store data from its URL analysis. The bottom diagram shows how Thug stores collected file samples with MongoDB's GridFS collection schema.

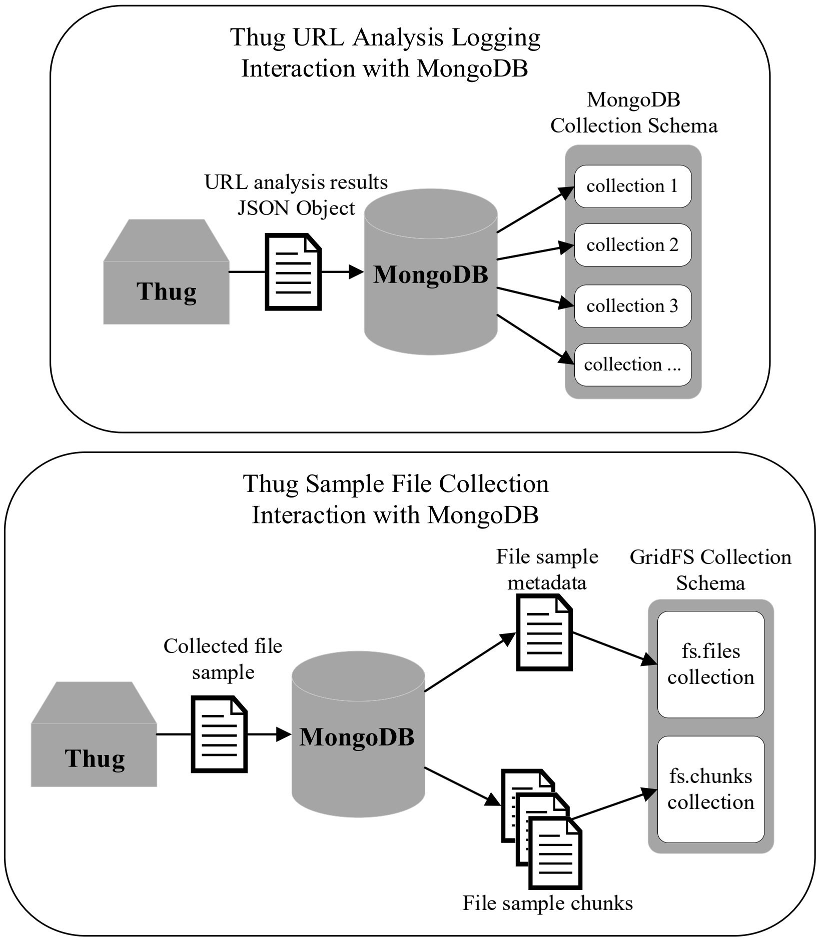


Figure 6. Thug interaction with MongoDB. Adapted from Mongo.org (2022)

Scrapy is an open-source Web-crawling and Web-scraping framework supporting data mining, monitoring, and automated testing (Scrapy.org, 2022). Scrapy is based on Twisted, a popular Python networking framework (Twisted Matrix Labs, 2022). Commercial Web crawlers like Googlebot (Google.com, 2022) collect information besides crawling IP addresses. However, we only need the secondary data sent by Web pages, not the pages themselves, and Scrapy sufficed. We also chose Scrapy due to its speed. We used Scrapy to crawl random IP addresses and record responses from Web servers for further analysis with Thug. The random IPv4 addresses were generated using the Python package Faker. The Web crawl flow chart is in Figure 7.

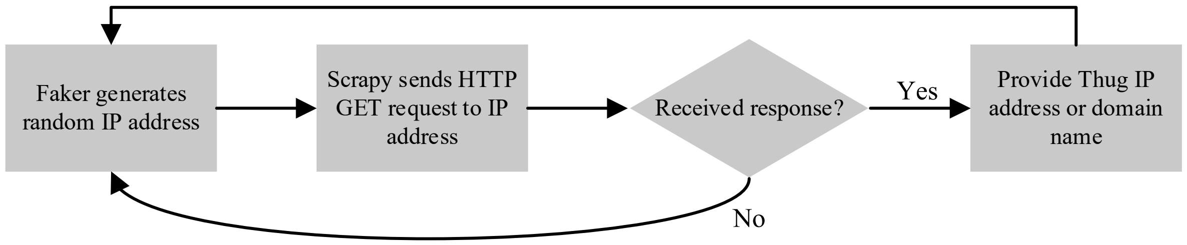

Faker generates random IP address
Scrapy sends HTTP GET request to IP address
Received response?
Yes
Provide Thug IP address or domain name
No

Figure 7. Web crawl flow chart

ClamAV is an open-source antivirus toolkit developed by Cisco Systems (Cisco Systems, 2022). ClamAV is quick and lightweight, and provides an easy-to-use interface from the command line. We used ClamAV to scan sample files downloaded from Thug's analysis to determine if they were malicious.

Additional URLs and IP addresses used for analysis included a sample of sites obtained from commercial threat intelligence vendors by our school. These sites had been "blacklisted" by various sources for sending data with known malicious signatures or otherwise showing suspicious behavior. However, they had not necessarily been observed to send drive-by downloads, and some may have been blacklisted for just hosting malicious users. Nonetheless, they were a richer source of malicious client activity than the random IP addresses we tested first.

C. FUNCTIONAL TESTING OF THUG

We tested the functions of Thug using public exploits and exploit tools. The Metasploit Framework is an open-source penetration-testing tool to find, exploit, and validate vulnerabilities (Metasploit.org, 2022). Metasploit provides modules for exploiting different operating systems, applications, and platforms. For this research, we used

Metasploit's browser-exploit modules to test Thug and assess its ability to identify browser and browser-plugin exploits used for drive-by downloads. We also tested some exploits not in Metasploit by coding the vulnerability as an HTML Web page on the testing server.

The Exploit Database collects publicly available exploits and its corresponding vulnerable software for vulnerability researchers and penetration testers (Offensive Security.org, 2022). It allows searching for exploits by title, CVE number, type, platform, and other parameters. It also provides exploits that were unavailable in Metasploit for the functional testing of Thug. We used the source code of these exploits to implement malicious HTML Web pages on our test server.

We tested Thug's ability to emulate different Web browser personalities using TCP/IP fingerprint analysis by NetworkMiner (Hjelmvik, 2022). NetworkMiner is an open-source network analysis tool that can conduct passive network-traffic analysis or analyze collected packets (Hjelmvik, 2022). NetworkMiner's analysis include extracting files or certificates from network traffic, decapsulation, and operating-system fingerprinting using the Satori and p0f databases (Hjelmvik, 2022). We used NetworkMiner to analyze packet-capture data from Thug's interaction with Web servers for Thug's personality assessment.

D. TEST ENVIRONMENT

We ran the Thug 4.0 client on a DigitalOcean cloud platform outside our campus network. DigitalOcean offers cloud computing services and virtualized resources. They call their Linux virtualized platform a "droplet." Our DigitalOcean machine ran the Oracle VirtualBox hypervisor to host a Linux virtual machine on which Thug and the supporting tools ran. The Thug virtual machine (VM) contained Thug, Scrapy, MongoDB, ClamAV, the malicious test server, and Metasploit, as in Figure 8. (The objects in red indicate a malicious Web server under our control for Thug's functional tests.) We connected to our servers through SSH and established SSH tunnels and virtual network computing (VNC) services for remote configuration and control. DigitalOcean does not implement firewall or intrusion-detection rules that could stop or impede HTTP responses to Thug. We also did not change the firewall configuration on our virtual machines and used the default settings. The default firewall rules block all unsolicited incoming traffic, allow unrestricted outgoing traffic, and only allow incoming traffic associated with outgoing communications.

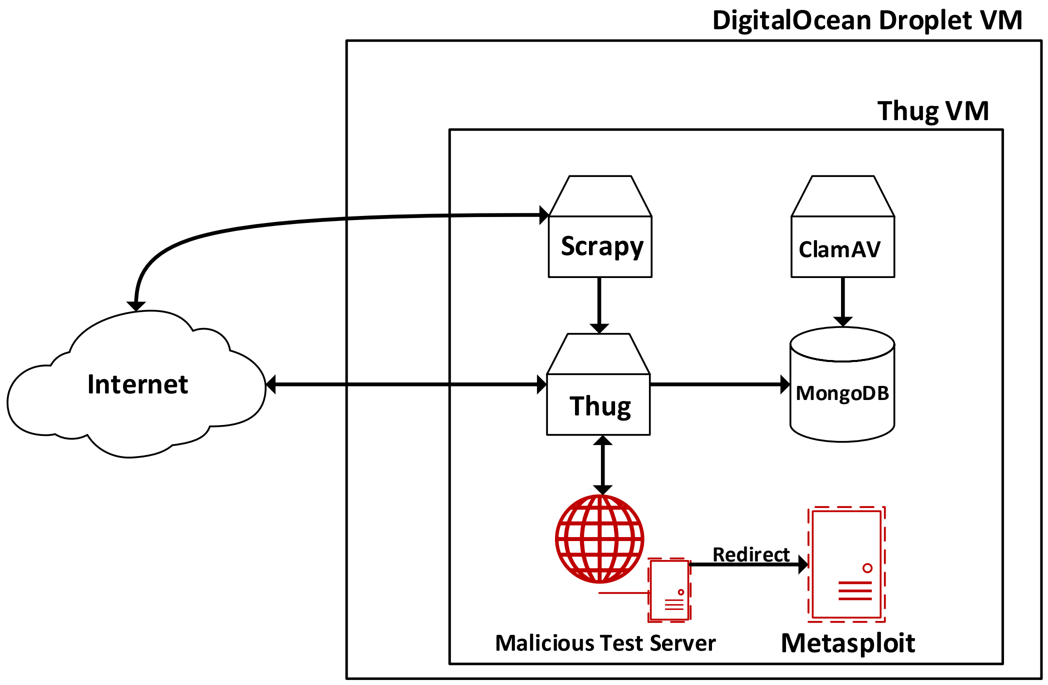


Figure 8. Test environment

We configured both machines with Ubuntu Server 20.04 LTS. Table 3 shows the DigitalOcean virtual machine hardware specifications, and Table 4 shows the hardware specifications for the Thug virtual machine.

Table 3. DigitalOcean Droplet virtual machine

Operating System	Ubuntu Server 20.04 LTS 64-bit
Memory	16 GB
Processor	4 vCPU (dedicated)
Disk	100GB SSD
Storage Volume	200 GB

Table 4. Thug virtual machine

Operating System	Ubuntu Server 20.04 LTS 64-bit
Memory	12 GB
Processor	4 vCPU
Disk	190GB vdi

E. EXPERIMENTATION PLAN

We ran three experiments with Thug's default configuration, emulating Windows XP running Internet Explorer 6.0 with Adobe Acrobat Reader 9.1.0, JavaPlugin 1.6.0.32, and Shockwave 10.0.64.0 plugins enabled. Experiment 1 was a functional test of Thug's ability to identify and classify drive-by exploits. This experiment analyzed a test Web server under our control. The server either ran the Metasploit modules by HTTP redirects or served HTML Web pages with malicious code from the Exploit Database. The Eicar antimalware test file served as the malicious payload for all the exploits (Eicar.org, 2022). Cybersecurity vendors use its signature to trigger test alerts with antivirus tools without the dangers of using actual malware. We selected the exploits based on the vulnerability identification modules in Thug's source code and several other exploits without corresponding modules. We assessed Thug's ability to identify which exploit was used in the drive-by download and successfully retrieved the anti-malware test file.

Our assessment criteria of Thug's performance were based on the amount of information Thug provided for each exploit.

- A *correct result* meant that Thug correctly identified the exploit by name or provided an associated CVE number.
- A *partially correct result* meant Thug did not identify the vulnerability by name but noted evidence of malicious activity by identifying the suspicious behavior or retrieving the exploit's malicious shell code. We used these clues to identify the exploit based on its signatures.

- A *nonfunctional result* meant we could not configure the exploit to work correctly, either through misconfiguration of the Metasploit software, the HTML code of the exploit, or an incompatible environment.

- An *incorrect result* meant that Thug could not identify the exploit or find an indicator of malicious activity.

Experiments 2 and 3 were a four-step process using Thug to analyze IP addresses. The first step found IP addresses or URLs with running Web servers. Experiment 2 used addresses identified using the Scrapy Web crawler to scan random IP addresses and check if a Web server was running on that machine; Experiment 3 used addresses obtained from a commercial blacklist. Experiments 2 and 3 both then fed the received URLs and IP addresses to Thug for its analysis. Then the data stored on MongoDB was reviewed for malicious activities and Thug's analysis. The final step extracted the collected sample files and scanned them for malware with the ClamAV antimalware tool, as in Figure 9. The obtained malware was then categorized based on the analysis by ClamAV.

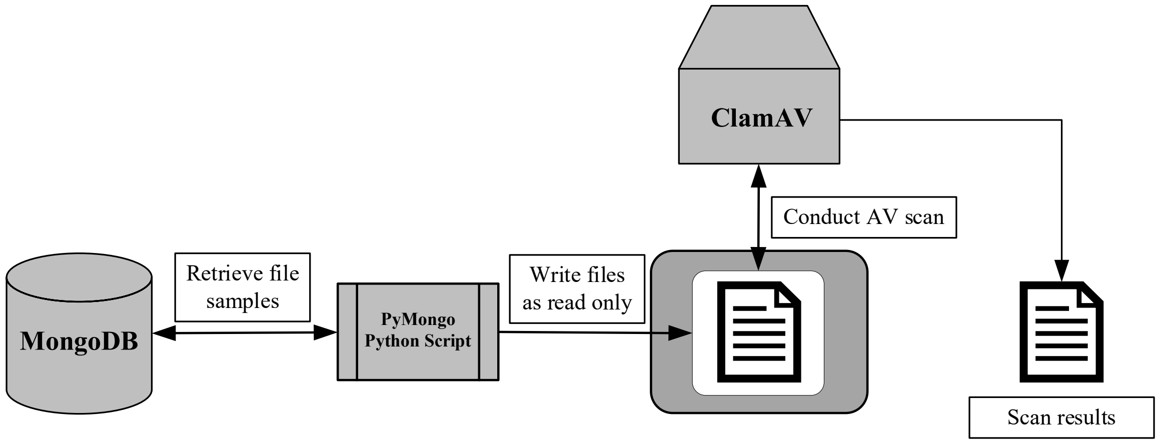


Figure 9. Malware analysis workflow

We prevented malware from infecting our machines through Thug's emulation and the inherent security measures in Unix-based operating systems, specifically Unix file permissions. Because Thug only emulates vulnerabilities, malicious servers cannot exploit the vulnerabilities with malicious shell code that could result in installing or executing the payload malware. The malicious servers do believe they are exploiting a vulnerability by sending a payload to execute. However, malicious shell code does not run; Thug collects the payload and stores it in MongoDB. The malware is inert because GridFS separates the files into metadata and chunks. Once the malware files are ready for analysis, a Python script using the PyMongo package retrieves the malware and stores them in a quarantined folder with read-only privileges. Writing files in read-only mode ensures that malicious binaries cannot execute.

IV. IMPLEMENTATION

A. THUG PERSONALITY ASSESSMENT

To assess Thug's ability to emulate different Web browsers and operating systems, we did TCP/IP fingerprint analysis during Thug's interaction with a Web server. This looked at specific TCP and IP header fields of the source traffic and matched them to a specific operating system. Since different operating systems use different default values for TCP fields such as time-to-live or window size, their values can identify the operating system. We selected five Thug "personalities" of operating systems and Web browsers for our TCP/IP fingerprint analysis, shown in Table 5. Personalities are specific operating systems combined with specific Web browsers. Thug claims personalities by setting user-agent fields in the HTTP GET request header when connecting to a Web server.

Table 5. Thug personalities selected for assessment. Adapted from Dell'Aera (2022)

winxpie60	Windows XP, Internet Explorer 6.0
win10ie110	Windows 10, Internet Explorer 11.0
osx11safari14	MacOS X 11.2.3, Safari 14.0.3
linuxchrome98	Linux, Chrome 98.0.4758.102
ipadchrome47	iPad iOS 9.1, Chrome 47.0.2526.70

We wanted to see if this emulation method would suffice to deceive a Web server using more sophisticated methods to identify connecting clients, such as TCP/IP fingerprinting. For this, we configured the DigitalOcean Droplet virtual machine to run Tcpdump to capture packet data between Thug and the Web server during their interaction, shown in Figure 10.

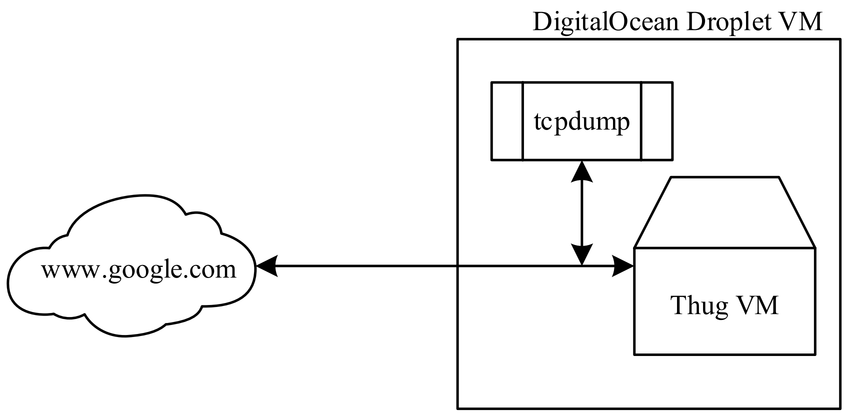


Figure 10. Personality assessment configuration

Tcpdump collected packets only on network traffic that made connections on port 80 or 443. We used Thug to analyze Google's Web-search homepage, due to its small HTTP footprint, to facilitate quick analysis by Thug. We this page for analysis because we did not anticipate seeing malicious behavior; rather, we were interested in observing the interaction between Thug and Google's Web server. We also did a control test using a Windows 11 laptop running the latest Chrome Web browser and used Wireshark to collect the network traffic data between the control browser and Google's Web server, as shown in Figure 11. We conducted the TCP/IP fingerprint analysis on the packet captures with NetworkMiner.

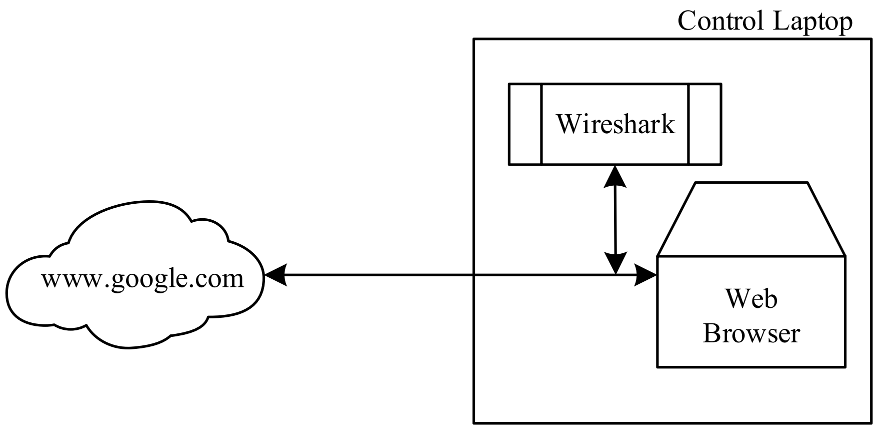
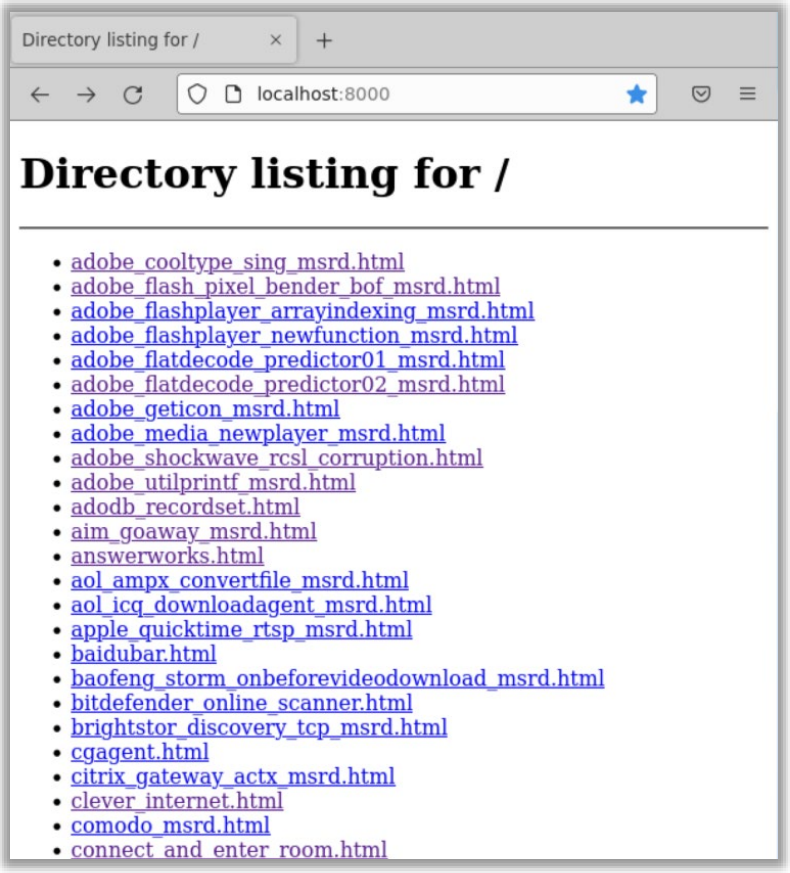


Figure 11. Personality assessment control configuration

B. EXPERIMENT 1: FUNCTIONAL ASSESSMENT

We assessed Thug's functions by configuring our malicious Web server to test its ability to identify and classify drive-by download exploits. We created a simple Web server using Python's built-in server, which can be started using the command `python3 -m http.server`; the command line argument `-m http.server` tells Python to run the script named `http.server`, which is our Web server. This Web server used "localhost" as the IP address and used port 8000 for HTTP access.

Python's Web server takes files from the directory in which the server was started and creates hyperlinks to all the documents in the directory to display on a webpage, shown in Figure 12. We accessed our Web server through the DigitalOcean Droplet virtual machine by forwarding port 8000 to the Thug virtual machine through VirtualBox's settings page. Web pages were manually created by saving files in the directory as HTML using Vim, a command-line text editor. We examined 107 vulnerability identification modules in Thug's source code and selected exploits we could find on Metasploit or The Exploit Database for testing. Also, we selected seven exploits without corresponding identification modules for testing to observe how Thug would classify exploits it was not explicitly programed to identify. Overall, we chose 99 exploits for testing (Appendix A).

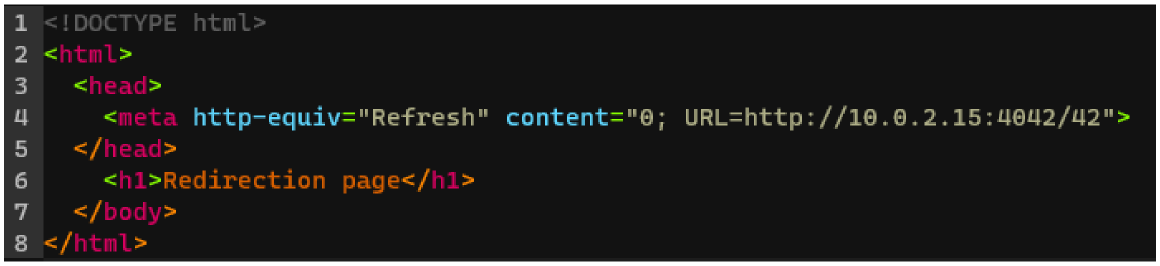


Directory listing for /

- adobe_cooltype_sing_msrd.html
- adobe_flash_pixel_bender_bof_msrd.html
- adobe_flashplayer_arrayindexing_msrd.html
- adobe_flashplayer_newfunction_msrd.html
- adobe_flatdecode_predictor01_msrd.html
- adobe_flatdecode_predictor02_msrd.html
- adobe_geticon_msrd.html
- adobe_media_newplayer_msrd.html
- adobe_shockwave_rcsl_corruption.html
- adobe_utilprintf_msrd.html
- adodb_recordset.html
- aim_goaway_msrd.html
- answerworks.html
- aol_ampx_convertfile_msrd.html
- aol_icq_downloadagent_msrd.html
- apple_quicktime_rtsp_msrd.html
- baidubar.html
- baofeng_storm_onbeforevideodownload_msrd.html
- bitdefender_online_scanner.html
- brightstor_discovery_tcp_msrd.html
- cgagent.html
- citrix_gateway_actx_msrd.html
- clever_internet.html
- comodo_msrd.html
- connect_and_enter_room.html

Figure 12. Screenshot of directory listing in the malicious test Web server

For simplicity and ease of deployment, we used Metasploit's browser exploit modules when possible. We installed Metasploit in its default configuration on the Thug virtual machine shown in Figure 8. After installation, we set the global payload option to `windows/download_exec` with the Eicar anti-malware test file as the payload. This payload option represents a malicious drive-by download, and the Eicar anti-malware test file represents malware. We wrote a startup script for Metasploit that loaded the 99 browser exploit modules we selected, and configured them with unique URI paths and port numbers. For each exploit module, we created a malicious HTML document in which the `http-equiv` parameter of the HTML "<meta>" tag was set to "Refresh" and the `content` parameter was set to the URL of a Metasploit module by `0; URL=http://10.0.2.15:40XX/XX` (Figure 13). These two commands force the client to do an HTML redirect, causing the visiting Web browser to refresh its page and load the page of the exploit on the Metasploit server.



```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Refresh" content="0; URL=http://10.0.2.15:4042/42">
  </head>
    <h1>Redirection page</h1>
  </body>
</html>
```

Figure 13. Metasploit redirect Web page

If a selected exploit was unavailable in Metasploit, we searched the Exploit Database for the HTML code of the exploit. We then coded the exploit into our Web server as its own Web page. If the webpage redirected to the Metasploit server, we included “msrd” at the end of the file name to indicate “Metasploit redirect.” If a file does not have this suffix, then the HTML document was coded using the exploit source code from The Exploit Database. In total, 56 Web pages were coded to redirect to Metasploit, and 43 were manually coded with malicious source code from The Exploit Database.

Once our malicious Web server was configured with exploits, we wrote a Python script to iterate through the HTML links on our Web server and invoke Thug analysis with a ten-second wait time between each link’s analysis to ensure Metasploit could keep up with the pace of Thug. After the analysis completed, we reviewed the results logged in MongoDB and assessed Thug’s ability to identify the exploit. We first checked the “exploits” collection in MongoDB (Table 2) to see if Thug had identified the exploit used by name or CVE number. If the exploit was not named outright, we reviewed the logs of the “behaviors” and “codes” collections for clues to identify the exploit. If the shell code of the exploit was logged in the “codes” collection, we conducted a reverse search of the shell code and identified the exploit. If the shell code was unavailable, we reviewed the “behaviors” collection and identified the exploit based on its observed behavior. Our assessment process is shown in Figure 14.

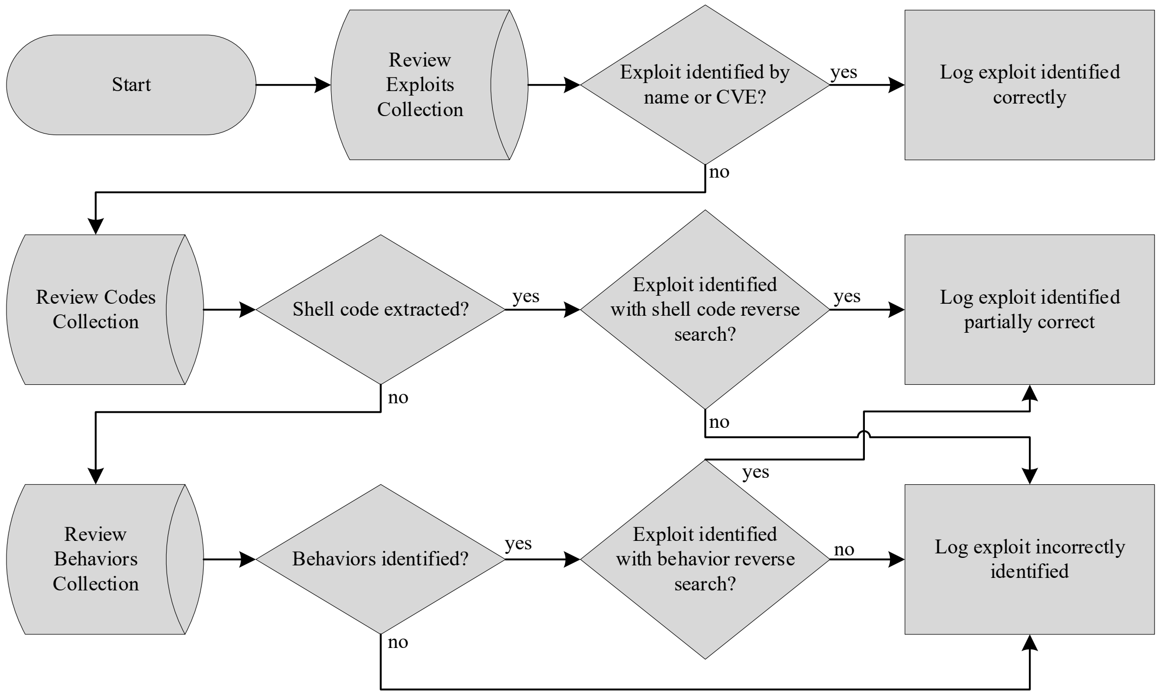

Start
Review Exploits Collection
Exploit identified by name or CVE?
yes
Log exploit identified correctly
no
Review Codes Collection
Shell code extracted?
yes
Exploit identified with shell code reverse search?
yes
Log exploit identified partially correct
no
no
yes
Review Behaviors Collection
Behaviors identified?
yes
Exploit identified with behavior reverse search?
no
Log exploit incorrectly identified
no

Figure 14. Thug functional test exploit identification workflow

During this experiment, Thug crashed while analyzing our HTML document containing the GlobalLink 2.7.0.8 ConnectAndEnterRoom ActiveX control stack buffer overflow exploit. Line 467 in Thug's source code "thug/thug/Logging/modules/MongoDB.py" returned a `TypeError: a byte-like object is required, not 'str'` error message during its analysis. Our debugging indicated that the error occurred when a string was returned to Thug's MongoDB logging function, which could be corrected by encoding the returned string into bytes. We reported the bug to Thug's author with our suggested fix, who promptly responded to our report and issued Fix #335 (Dell'Aera & Foley, 2022). We did not see any other discrepancies with Thug's operation.

C. EXPERIMENTS 2 AND 3: REAL-WORLD ATTACKS

Experiments 2 and 3 used Thug to analyze real-world Web servers for client-side attacks. In Experiment 2, source sites were collected by randomly crawling IP addresses and looking for Web servers running on those machines. We wrote a Python program that used the Python package Faker to generate 6,000,000 random valid IPv4 addresses,

addresses are not in the private or reserved IP address space. A script then crawled the addresses with Scrapy and recorded whether the IP addresses responded to an HTTP GET request. Any response meant the host machine was running a Web server. For this collection, we configured Scrapy's settings to support 5,000 concurrent requests and set the thread pool max size to 5,000, with a download timeout of 15 seconds and a depth priority of one. These settings provided excellent crawling performance.

If a response from a Web server was received, its domain name or IP address was recorded and added to a list. Out of the 6,000,000 generated IP addresses, our crawler identified 37,415 as running Web servers. We wrote a Python script to iterate through the IP addresses and domain names and invoke Thug to analyze each entry. We used Thug's default personality, Windows XP with Internet Explorer 8.0, with no other options passed to the command line. Experiment 3 was conducted similarly except that the source sites were obtained from a commercial blacklist containing 83,667 IP addresses and URLs.

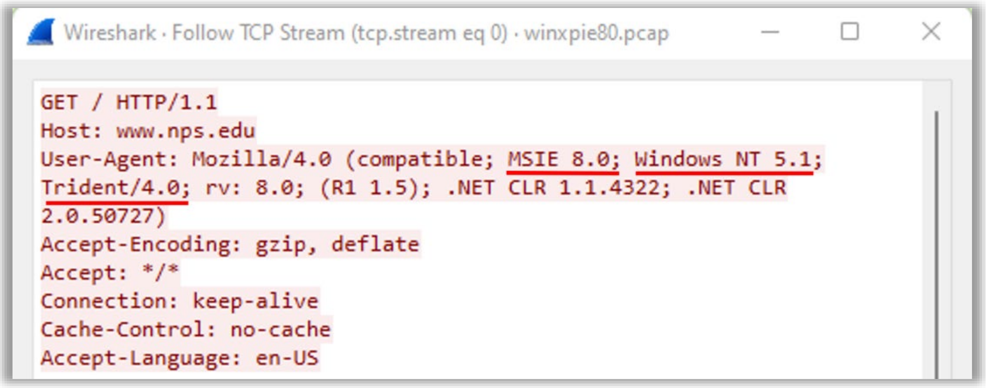
After Thug's analysis was complete, we reviewed the log data in MongoDB and recorded the number of exploits identified. We used MongoDB's Mongosh interface to access the database for our review, with particular attention to the "exploits," "codes," "behaviors," and "locations" collections (Table 2). Next, we extracted the collected file samples stored in the MongoDB with GridFS using a Python program with PyMongo. The program iterates through MongoDB's fs.files collection, and then uses the GridFS package to reassemble the files and export them to a directory. The MongoDB engineering team aided us during this process. Once the file extraction was complete, we ran ClamAV with options to identify malware infected files and potentially unwanted applications (PUA). Any file identified as PUA or infected with malware were quarantined to a separate folder. The remaining files were deleted.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RESULTS AND DISCUSSION

A. THUG PERSONALITY ASSESSMENT RESULTS

The personality assessment showed that TCP/IP fingerprint analysis with NetworkMiner could identify Thug's underlying operating system with 50% confidence. When we examined the packet capture from the assessment, we observed that the HTTP user-agent fields are consistent with the chosen personalities from Table 5. Figure 15 shows the TCP stream for the HTTP GET request with the appropriate user-agent field for Windows XP running Internet Explorer 8.0.


Wireshark · Follow TCP Stream (tcp.stream eq 0) · winxpie80.pcap

```
GET / HTTP/1.1
Host: www.nps.edu
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1;
Trident/4.0; rv: 8.0; (R1 1.5); .NET CLR 1.1.4322; .NET CLR
2.0.50727)
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cache-Control: no-cache
Accept-Language: en-US
```

Figure 15. Wireshark TCP stream of Thug's HTTP GET request

However, in the packet captures for all five tests, we observed that the default TTL (time to live) values and TCP window sizes were inconsistent with the target emulated operating systems. For example when emulating Windows XP, Thug uses a TTL value of 64 and a TCP window size of 64240, while Windows XP's default values are 128 and 65535 respectively (Hjelmvik, 2011). Thug used the same values of these two parameters for all five tested personalities we selected (Table 5) during the analysis. Using these fingerprints and the Satori OS fingerprint database, NetworkMiner identified Thug with a 50% confidence level as running on a Linux-based operating system, with Windows 10 as the other possibility. For a control test using the Windows 11 laptop with the latest Chrome browser, NetworkMiner identified the laptop as a Windows-based system with 100%

confidence using the P0f database, and as a Windows 10 system with 85.71% confidence using the Satori database. We consider NetworkMiner's identification as close to correct for the control test, as Windows 11 is primarily based on Windows 10 (Stewart et al., 2022). The summary of our assessment results is shown in Table 6.

Table 6. Thug personality assessment results

Personality	NetworkMiner identification
winxpie60	Linux 5.4 (50%) Windows 10 (50%)
win10ie110	Linux 5.4 (50%) Windows 10 (50%)
osx11safari14	Linux 5.4 (50%) Windows 10 (50%)
linuxchrome98	Linux 5.4 (50%) Windows 10 (50%)
ipadchrome47	Linux 5.4 (50%) Windows 10 (50%)
Windows 11 control	Windows (100%): Windows 10 (85.71%)

B. EXPERIMENT 1 RESULTS

Experiment 1 showed that Thug could identify the malicious artifacts of our Web server. Overall, Thug recognized 85 malicious exploits of 99. Thug correctly identified 45 by name or provided a CVE number. It also identified another 40 as malicious from general behavior observations such as ActiveX control abuse, malicious pixel iframes, or malicious shell code of the exploit. A summary of Thug's analysis on 84 substantially correct cases is shown in Table 7. "AA?" indicates that additional analysis was needed to identify the exploit based on signatures from extracted shell code or from malicious behavior, marked as "SC" and "MB," respectively.

Table 7. Thug’s correct and partially correct results on Experiment 1

Exploit	AA?	Exploit	AA?	Exploit	AA?	Exploit	AA?
AIM goaway CVE: 2004-0636	MB	AOL Radio AmpX CVE: N/A		Adobe Flash copyPixels CVE: 2014-0556	MB	IBM Lotus Notes Client CVE: 2012-2174	SC
Adobe ‘Collab.getIcon()’ CVE: 2009-0927	MB	Adobe ‘Doc.media.newPlayer’ CVE: 2009-4324	MB	Java 7 Applet CVE: 2012-4681	MB	ADODB.Recordset CVE: 2006-3354	
Adobe Flash Player ‘newfunction’ CVE: 2010-1297	MB	Adobe ‘util.printf()’ CVE: 2008-2992	MB	AnswerWorks CVE: 2007-6387		Baidu Search Bar CVE: 2007-4105	SC
Creative Software AutoUpdate Engine CVE: 2008-0955	SC	MS DirectShow ‘msvidctl.dll’ CVE: 2008-0015	SC	BitDefender Online Scanner CVE: 2007-6189		ChinaGames ‘CGAgent.dll’ CVE: 2009-1800	
IBM Lotus Domino DWA Upload Module CVE: 2007-4474		IBM Lotus ‘inotes6.dll’ CVE: 2007-4474	SC	GlobalLink 2.7.0.8 CVE: 2007-5722	SC	DivX Player 6.6.0 CVE: 2008-0090	SC
EnjoySAP SAP GUI CVE: 2008-4830		Facebook Photo Uploader CVE: 2008-5711		D-Link Audio Control CVE: 2008-4771		Xunlei Web Thunder CVE: 2007-5064	SC
Gateway WebLaunch CVE: 2008-0220	SC	GOM Player CVE: 2007-5779		Lycos FileUploader CVE: 2008-0443	SC	Ourgame ‘GLIEDown2.dll’ CVE: N/A	
ICQ Toolbar CVE: 2008-7136		MS14-064 CVE: 2014-6342	MB	HP Compaq Notebooks CVE: 2007-6333		Clever Internet CVE: 2007-4067	SC
MSXML Memory Corruption CVE 2012-1889	SC	Macrovision Installshield CVE: 2007-5660		Java Deployment Toolkit CVE: 2010-0886		jetAudio 7.x CVE: 2007-4983	
Macrovision FlexNet CVE: 2008-4586		MS IE XML CVE: 2006-5745		Move Networks CVE: 2008-1044		MS Rich Textbox CVE: 2008-0237	SC
NCTAudioFile2 CVE: 2007-0018		RealPlayer ‘ierpplug.dll’ CVE:2007-5601	SC	MySpace Uploader CVE: 2008-0659		Sejoong Namo CVE: 2008-0634	
Apple QuickTime CVE: 2007-6166	SC	Shockwave rcsL CVE: 2010-3653	SC	NeoTracePro 3.25 CVE: 2006-6707	SC	Nessus Delete File CVE: 2007-4031	
MS Silverlight CVE: 2013-3896	MB	Microsoft Access CVE: 2008-2463		Nessus Command Execution CVE: 2007-4062		Office Viewer OCX CVE: 2007-2588	SC
SonicWALL NetExtender CVE: 2007-5603		MS OWC Spreadsheet CVE: 2009-1534	SC	Xunlei XPPlayer CVE: N/A	SC	Cisco Linksys PTZ Cam CVE: 2012-0284	SC
BaoFeng Storm CVE: 2009-1612		Symantec AppStream CVE: 2008-4388		Quantum Streaming Player CVE: 2008-1044		Qvod Player 2.1.5 CVE: 2008-4664	SC
Symantec BackupExec CVE: 2007-6016		MS Visual Studio CVE: 2008-3704	SC	Rediff Bol Downloader CVE: 2006-6838	SC	Rising Scanner CVE: N/A	SC
MS Media Encoder CVE: 2008-3008		MS Internet Explorer Unsafe Scripting CVE: N/A		Sina DLoader Class CVE: 2008-6442		StreamAudio Chaincast CVE: 2008-0248	
MS IE WebViewFolderIcon CVE: 2006-3730		WinZip FileView CVE: 2006-5198		Toshiba Surveillance CVE: 2008-0399		UUSee ‘Update’ CVE: 2008-7168	SC
Winamp Playlist UNC Path CVE: 2006-0476	SC	HP LoadRunner CVE: 2007-6530		VLC Remote Bad Pointer CVE: 2007-6262		Firefox ‘WMP’ CVE: 2010-2745	SC
Yahoo! Messenger CVE: 2007-4515		Zenturi ProgramChecker CVE: 2007-2987		MS IE Remote Wscript CVE: 2004-0549		Yahoo! JukeBox CVE: 2008-0625	
Adobe CoolType CVE: 2010-2883	MB	Adobe Flash Player CVE: 2011-2110	MB	Yahoo! Messenger CVE: 2007-5017	SC	Yahoo! Messenger ‘YVerinfo.dll’ CVE: 2007-4515	

In 13 other cases, the exploit failed to deposit malicious files on our victim machine due to either misconfiguration of the exploit or an incompatible environment. These cases are summarized in Table 8. Most were caused by an incompatible Metasploit module or incompatible environment, though we could not identify the cause of failure to execute the exploit in four cases. Thug did not identify the malicious nature of an exploit at all in one case, Ourgame GL World 2.x with CVE number 2008-0647. Thug's analysis of this particular exploit did not log an exploit name or CVE number in the "exploits" collection, did not extract and identify shell code in the "codes" collection, or identify any malicious behaviors in the "behavior" collection. It appears that Thug keeps up to date with obfuscations and other deception associated with drive-by downloads. Statistics of Experiment 1 are shown in Table 9.

Table 8. Thug nonfunctional results

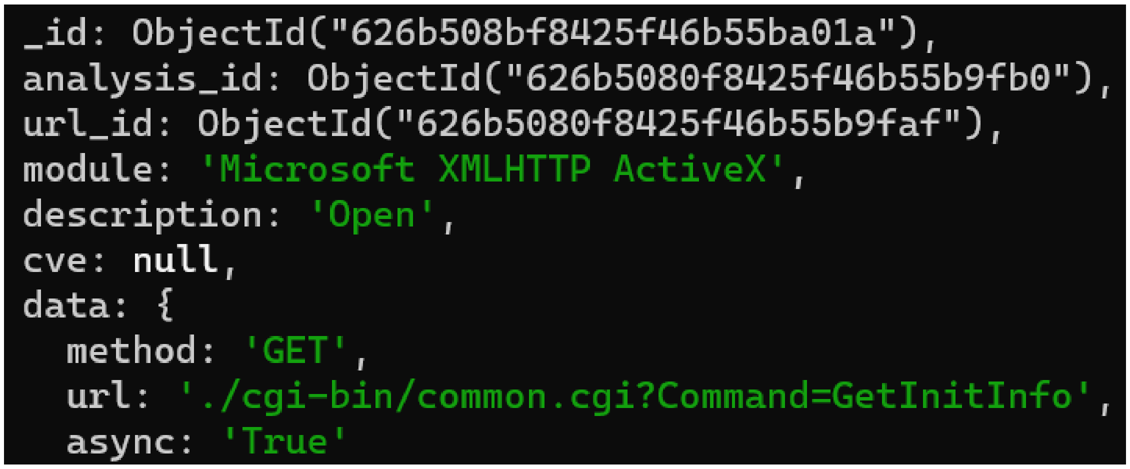
Exploit	Failure reason
Adobe flatedecode predictor 01 CVE: 2009-3459	Unknown
Adode flatedecode predictor 02 CVE: 2009-3459	Incompatible exploit module
CA BrightStor Discovery Service CVE: 2005-2535	Incompatible exploit module
Comodo credential gatherer CVE: N/A	Incompatible environment
MS DirectX DirectShow CVE: 2007-3901	Incompatible exploit module
Microsoft Works 7 CVE: 2008-1898	Incompatible exploit module
MS IE XMLDOM filename disclosure CVE: 2013-7331	Incompatible environment
Microsoft IIS RDS DataStub CVE: 2002-1142	Incompatible exploit module
Kaseya Virtual System Administrator CVE: 2015-6922	Incompatible exploit module
HP Easy Printer Care CVE: 2011-4786	Unknown
Honeywell HSC Remote Deployer CVE: 2013-0108	Unknown
iMesh 7 'IMWebControl' CVE: 2007-6493	Unknown
KingSoft 'UpdateOcs2.dll' CVE: 2008-1307	Compilation error

Table 9. Thug functional assessment results summary

Result Type	Frequency
Correct	45
Partially correct	40
Incorrect	1
Non-functional	13

C. EXPERIMENT 2 RESULTS

Experiment 2 directed Thug to test random IP addresses. As expected, most results were uninteresting since malicious sites are statistically rare. Minor anomalies of these sites were flagged on occasion. Overall, Experiment 2 analyzed 37,415 Web sites. Thug identified 2,054 Web servers with suspicious behavior, all being abuse of ActiveX control GET and POST methods. An example of Thug's analysis results is shown in Figure 16. In this example, an ActiveX GET method is used to pull content from a secondary site.



```
_id: ObjectId("626b508bf8425f46b55ba01a"),
analysis_id: ObjectId("626b5080f8425f46b55b9fb0"),
url_id: ObjectId("626b5080f8425f46b55b9faf"),
module: 'Microsoft XMLHTTP ActiveX',
description: 'Open',
cve: null,
data: {
  method: 'GET',
  url: './cgi-bin/common.cgi?Command=GetInitInfo',
  async: 'True'
```

Figure 16. Thug report of malicious behavior

For this analysis, 146,768 file samples were downloaded, of which ClamAV identified 18 as infected with malware and 230 as potentially unwanted applications (PUA). For Experiments 2, we classified the malware into 12 categories based on ClamAV's signature names, as shown in Figure 17. ClamAV's signature names follow the format “{platform}.{category}.{name}-{signature id}-{revision},” where platform denotes the operating system of the malware, category describes the malware type, and then the name, signature identification and revision number follow (Cisco Systems Inc., 2022). Most malware fell into eight distinct categories based on the platform and category description, the bottom eight in Figure 17. Malware listed in the “Other” category included malware targeted at specific applications such as Microsoft Office document macros, HTML, and Java applications. Figure 18 shows the percentage totals of the malware by type.

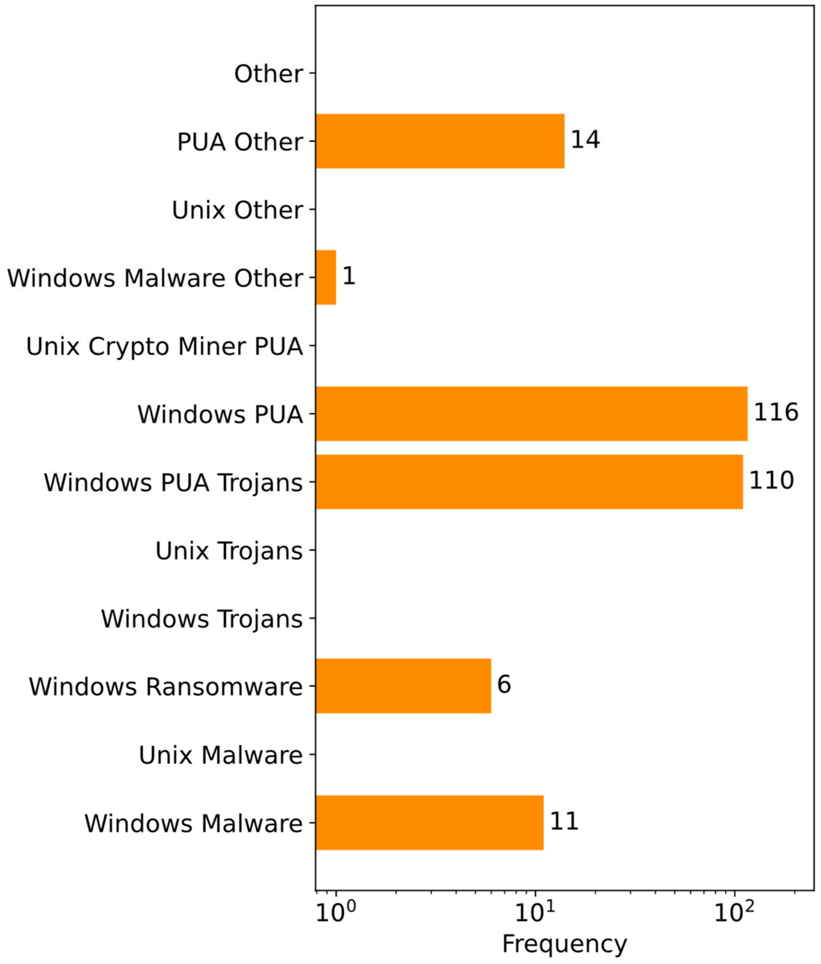


Figure 17. Frequency of malware downloaded from random Web servers by malware type

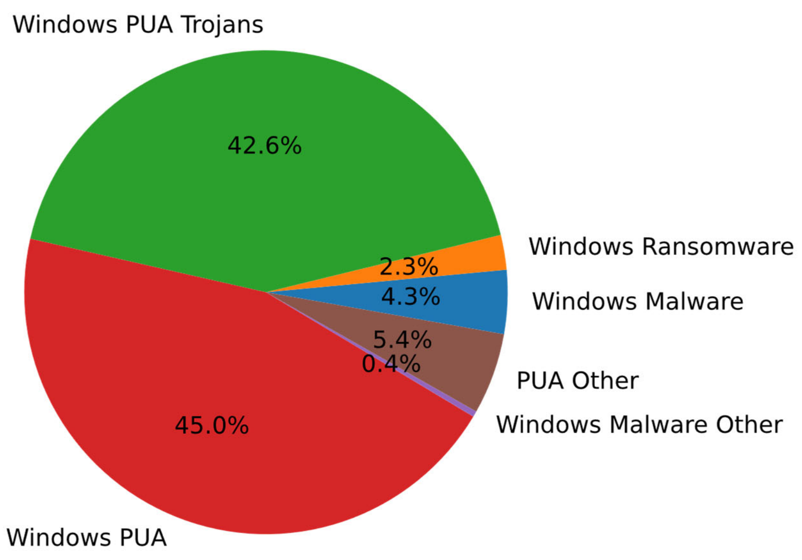


Figure 18. Proportion of malware downloaded from random Web servers by type

The PUAs identified were trojans, adware, and trackers. An example that ClamAV identified was the "PUA.Html.Trojan.Agent-37074"; a VirusTotal scan showed that 24 of 61 antivirus tools flagged this file as malicious. Microsoft and Trend Micro security blogs identified the PUA as Exploit:HTML/Phominer.A and Trojan.HTML.IFRAME.FASGU respectively (Microsoft, 2017; Fuentebella, 2022). Their reports indicate this PUA is dropped onto victim computers when visiting malicious Web sites. It can steal information from the victim's computer, and can embed malicious Iframes to redirect users to other malicious sites.

D. EXPERIMENT 3 RESULTS

Experiment 3 used Thug to analyze and collect data from a commercial blacklist of IP addresses and domain names. We obtained this list from the Information Technology department at our School; the list included sites that were malicious for many reasons, not necessarily for drive-by downloads. As in Experiment 2, we also used the same 12 ClamAV malware categories. The results of this experiment were more interesting than those of Experiment 2, with fewer observed malicious behaviors in website interaction (e.g., content delivery by ActiveX) but more kinds of malware. We analyzed 83,667 Web servers, of which Thug identified 953 with malicious activity. As with Experiment 2, all malicious behaviors observed used ActiveX control abuse with GET and POST methods. During the analysis, Thug collected 602,731 file samples, of which ClamAV identified 2,043 as malware and an additional 869 as PUAs; statistics are shown in Figure 19 and Figure 20.

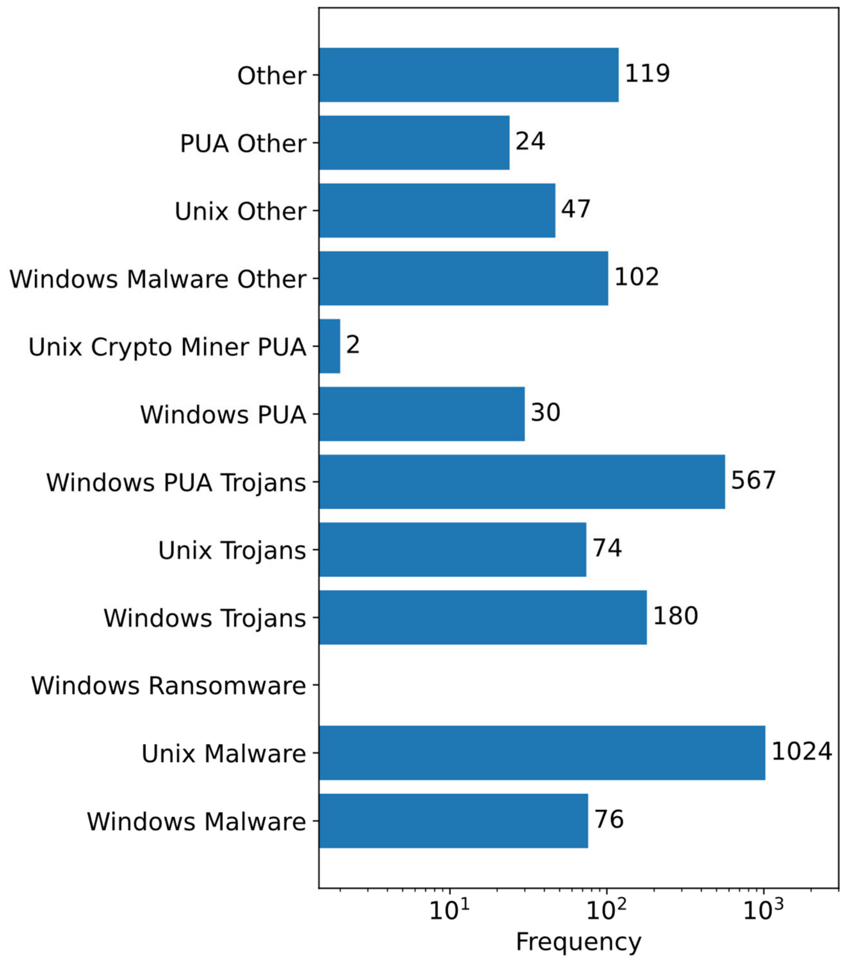


Figure 19. Frequency of malware downloaded from blacklist Web servers by type

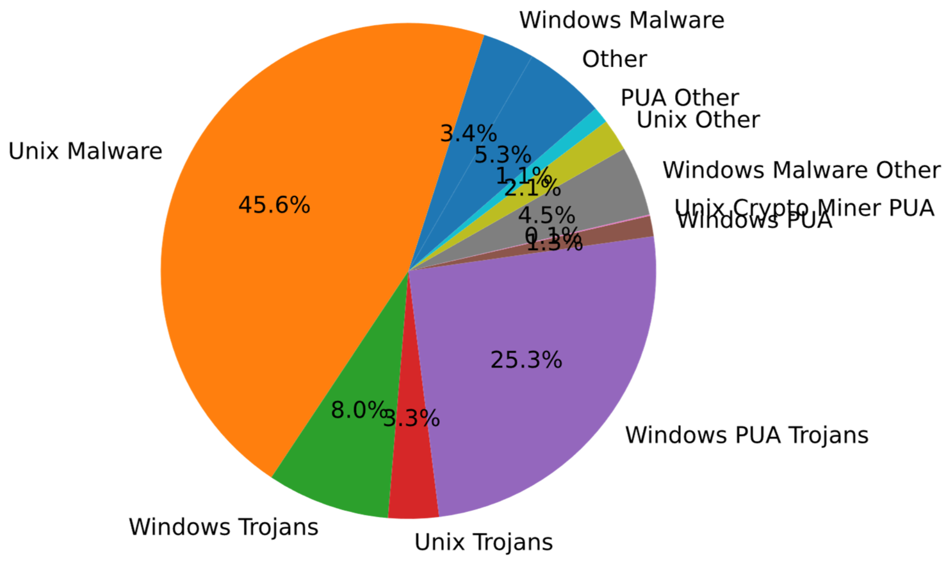


Figure 20. Proportion of malware downloaded from blacklist Web servers by type

The fewer observed malicious behavior may indicate that the blacklist only contains Web servers with malicious payloads, not those with exploits used to start the drive-by download. Table 10 shows the results of Experiments 2 and 3. None of the random Web servers scanned in Experiment 2 were in the blacklist for Experiment 3.

Table 10. Results from Experiments 2 and 3

	Sites	**Malicious Behavior**	**Malware**	**PUA**
Blacklist	83,667	953 (1.14%)	2,043 (2.44%)	869 (1.03%)
Random Web	37,415	2,045 (5.47%	18 (0.04%)	230 (0.61%)

E. DISCUSSION

Thug's personality assessment shows a sufficient ability to deceive malicious Web servers, but it can be improved. While the TCP/IP fingerprint may be adequate to deceive a Web server into thinking a system is a Windows or Linux-based operating system, it may not suffice for emulating MacOS, Android, or iOS-based operating systems. Improvements to Thug should be made by altering the default values of the TTL and TCP/IP window fields to those consistent with the known values of the emulated operating systems.

Although Thug performed well in categorizing and identifying different drive-by exploits on our malware server in Experiment 1, we did not observe as many exploits from the blacklist as from random IP addresses. Our results also indicated that drive-by downloads using methods besides ActiveX controls are rare.

The malware we collected did vary considerably. Experiment 2 provided 13 unique malware files, while Experiment 3 provided 163 unique malware files. In Experiment 2, 5.4% of random Web servers showed signs of malicious activity, which is similar to previous experiments (Ikinci et al., 2008; Keats et al., 2007; Qassrawi & Zhang, 2011). However, not all of the 2,054 malicious interactions downloaded and executed malware. This observed behavior suggests that many malicious activities find it more profitable to collect reconnaissance for future use. Our suspicion is further supported by the fact that the random Web servers delivered more PUAs than malware files, as PUAs often are spyware, adware, or trackers.

Experiment 3 showed interesting results In the significant difference between malicious behaviors observed and malware downloaded, likely due to the types of servers included in the blacklist, which typically host the malware payload instead of pulling contents from other malicious servers. Legitimate Web servers may be compromised by malicious advertisements (malvertising) or attackers embedding malicious redirects. The goal of the blacklist is to prevent the malware from reaching the unsuspecting user without limiting access to legitimate websites. The comparatively fewer PUAs may indicate that blacklisted Web servers have more ambitious malicious intentions than most Web servers.

Comparisons of the malware (Figure 21) collected between Experiments 2 and 3 show interesting results. The only ransomware seen was from the random Web servers used in Experiment 2. Ransomware is particularly damaging to victims, and our random Web server sample collected most PUAs with few instances of serious ransomware.

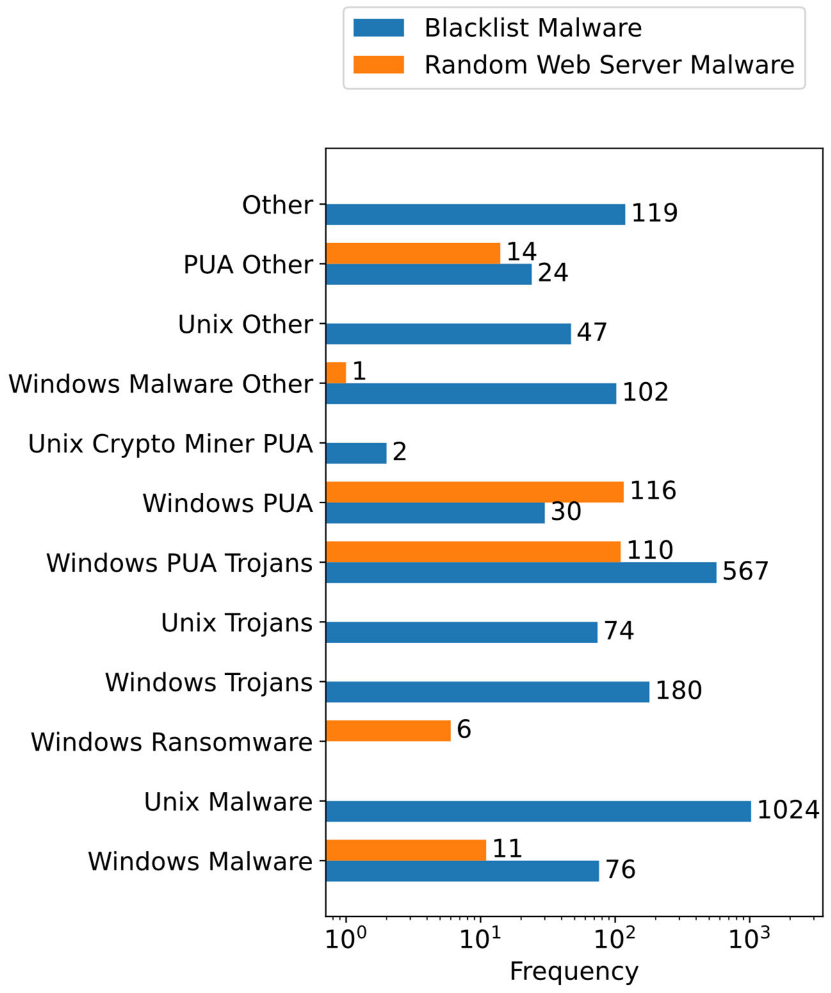


Figure 21. Comparison of malware frequency between Experiments 2 and 3.

Another interesting observation was the differences in the targeted operating system between the two experiments. Experiment 2 showed collected malware almost entirely targeted Windows operating systems, but most malware collected in Experiment 3 targeted Unix and Linux systems. Our test environment used a Linux-based operating system to host Thug, but we configured Thug to emulate a Windows XP system. The large occurrence of Unix-targeted malware suggests that the blacklisted web servers may be using more sophisticated techniques to identify host operating systems, while the malicious web servers from the random sample do not. This also argues for improvements to Thug's personality emulation.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION AND FUTURE WORKS

A. SUMMARY OF FINDINGS

Our personality assessment of Thug's emulation abilities showed TCP/IP fingerprint analysis can identify Thug's underlying operating system with a medium level of confidence. While Thug may fool some malicious Web servers, it might not fool more sophisticated Web servers using less-known methods to identify clients.

Experiment 1 demonstrated Thug's ability to identify and log a substantial number of known drive-by exploits. Although some of Thug's results required additional analysis to identify the exploit, Thug performed well overall during its functional testing, demonstrating its usefulness as a cybersecurity tool. Experiment 2 used Thug to analyze real-world Web servers from randomly generated IP addresses. We did not observe any interesting drive-by exploits, but we did collect a significant amount of malware. While most of the collected malware was classified as potentially unwanted applications such as spyware, adware, or trackers, the rest were generic with minor occurrences of ransomware. Experiment 3 used Thug to analyze real-world Web servers using addresses obtained from a commercial blacklist. Although we did not find any interesting drive-by exploits, we observed less malicious behavior with a much larger variation in the types of collected malware.

We demonstrated methods for finding and collecting malware for analysis using client-side honeypots such as Thug. We tested Thug's functionality, and our experiments confirmed its usefulness in collecting empirical data in a real-world scenario. Its benefit can only be best seen for the relatively rare malicious sites since it finds many uninteresting anomalies on randomly chosen sites. We conclude that client-side honeypots can provide added value to standard anti-malware tools.

B. FUTURE WORK

We recommend changing the TCP time-to-live and TCP window size values to those consistent with the target emulated operating system and repeating Experiment 3 on the blacklist Web servers to observe if a difference occurs in the amount of collected malware targeting Linux-based operating systems. Repeating the experiment with these changes could identify what methods malicious Web servers use to identify connecting clients. Additionally, for Thug's personality tests, we only used Windows XP and Internet Explorer 6.0. Repeating Experiments 2 and 3 using the same source list of IP addresses but with other personalities and comparing the differences in malicious behavior or delivered malware could reveal interesting insights into the operation of malicious Web servers.

Experiments 2 and 3 yielded much log data. Further analysis of it could provide insight to trends in the types of malware collected or the interactions between the client and Web server that led to the download of the malware. Such discoveries could be accomplished by extracting features from our dataset and using machine-learning algorithms to identify correlations.

APPENDIX A. EXPLOITS SELECTED FOR EXPERIMENT 1

This appendix lists the exploits we selected for testing in Experiment 1. The exploit column contains the name and applicable Mitre CVE number. "Imp." represents implementation method, "HTML" for manually coding the exploit or "MS" for Metasploit.

Table 11. Exploits selected for Experiment 1

Exploit	Imp.	Exploit	Imp.
AIM goaway CVE: 2004-0636	MS	Ourgame GL World 2.x with CVE number 2008-0647	HTML
Adobe 'Collab.getIcon()' CVE: 2009-0927	MS	Java Deployment Toolkit CVE: 2010-0886	HTML
Adobe Flash Player 'newfunction' CVE: 2010-1297	MS	Move Networks CVE: 2008-1044	HTML
Creative Software AutoUpdate Engine CVE: 2008-0955	MS	MySpace Uploader CVE: 2008-0659	HTML
IBM Lotus Domino DWA Upload Module CVE: 2007-4474	MS	NeoTracePro 3.25 CVE: 2006-6707	HTML
EnjoySAP SAP GUI CVE: 2008-4830	MS	Nessus Command Execution CVE: 2007-4062	HTML
Gateway WebLaunch CVE: 2008-0220	MS	Xunlei XPPlayer CVE: N/A	HTML
ICQ Toolbar CVE: 2008-7136	MS	Quantum Streaming Player CVE: 2008-1044	HTML
MSXML Memory Corruption CVE 2012-1889	MS	Rediff Bol Downloader CVE: 2006-6838	HTML
Macrovision FlexNet CVE: 2008-4586	MS	Sina DLoader Class CVE: 2008-6442	HTML
NCTAudioFile2 CVE: 2007-0018	MS	Toshiba Surveillance CVE: 2008-0399	HTML
Apple QuickTime CVE: 2007-6166	MS	VLC Remote Bad Pointer CVE: 2007-6262	HTML
MS Silverlight CVE: 2013-3896	MS	MS IE Remote Wscript CVE: 2004-0549	HTML
SonicWALL NetExtender CVE: 2007-5603	MS	Yahoo! Messenger CVE: 2007-5017	HTML
BaoFeng Storm CVE: 2009-1612	MS	IBM Lotus Notes Client CVE: 2012-2174	HTML
Symantec BackupExec CVE: 2007-6016	MS	ADODB.Recordset CVE: 2006-3354	HTML
MS Media Encoder CVE: 2008-3008	MS	Baidu Search Bar CVE: 2007-4105	HTML
MS IE WebViewFolderIcon CVE: 2006-3730	MS	ChinaGames 'CGAgent.dll' CVE: 2009-1800	HTML
Winamp Playlist UNC Path CVE: 2006-0476	MS	DivX Player 6.6.0 CVE: 2008-0090	HTML
Yahoo! Messenger CVE: 2007-4515	MS	Xunlei Web Thunder CVE: 2007-5064	HTML
Adobe CoolType CVE: 2010-2883	MS	Ourgame 'GLIEDown2.dll' CVE: N/A	HTML
AOL Radio AmpX CVE: N/A	MS	Clever Internet CVE: 2007-4067	HTML

Exploit	Imp.	Exploit	Imp.
Adobe ‘Doc.media.newPlayer’ CVE: 2009-4324	MS	jetAudio 7.x CVE: 2007-4983	HTML
Adobe ‘util.printf()’ CVE: 2008-2992	MS	MS Rich Textbox CVE: 2008-0237	HTML
MS DirectShow ‘msvidctl.dll’ CVE: 2008-0015	MS	Sejoong Namo CVE: 2008-0634	HTML
IBM Lotus ‘inotes6.dll’ CVE: 2007-4474	MS	Nessus Delete File CVE: 2007-4031	HTML
Facebook Photo Uploader CVE: 2008-5711	MS	Office Viewer OCX CVE: 2007-2588	HTML
GOM Player CVE: 2007-5779	MS	Cisco Linksys PTZ Cam CVE: 2012-0284	HTML
MS14-064 CVE: 2014-6342	MS	Qvod Player 2.1.5 CVE: 2008-4664	HTML
Macrovision Installshield CVE: 2007-5660	MS	Rising Scanner CVE: N/A	HTML
MS IE XML CVE: 2006-5745	MS	StreamAudio Chaincast CVE: 2008-0248	HTML
RealPlayer ‘ierpplug.dll’ CVE:2007-5601	MS	UUSee ‘Update’ CVE: 2008-7168	HTML
Shockwave RCSL CVE: 2010-3653	MS	Firefox ‘WMP’ CVE: 2010-2745	HTML
Microsoft Access CVE: 2008-2463	MS	Yahoo! JukeBox CVE: 2008-0625	HTML
MS OWC Spreadsheet CVE: 2009-1534	MS	Yahoo! Messenger ‘YVerinfo.dll’ CVE: 2007-4515	MS
Symantec AppStream CVE: 2008-4388	MS	Adobe flatedecode predictor 01 CVE: 2009-3459	MS
MS Visual Studio CVE: 2008-3704	MS	Adode flatedecode predictor 02 CVE: 2009-3459	MS
MS Internet Explorer Unsafe Scripting CVE: N/A	MS	CA BrightStor Discovery Service CVE: 2005-2535	MS
WinZip FileView CVE: 2006-5198	MS	Comodo credential gatherer CVE: N/A	MS
HP LoadRunner CVE: 2007-6530	MS	MS DirectX DirectShow CVE: 2007-3901	MS
Zenturi ProgramChecker CVE: 2007-2987	MS	Microsoft Works 7 CVE: 2008-1898	MS
Adobe Flash Player CVE: 2011-2110	MS	MS IE XMLDOM filename disclosure CVE: 2013-7331	MS
Adobe Flash copyPixels CVE: 2014-0556	MS	Microsoft IIS RDS DataStub CVE: 2002-1142	MS
Java 7 Applet CVE: 2012-4681	MS	Kaseya Virtual System Administrator CVE: 2015-6922	MS
AnswerWorks CVE: 2007-6387	HTML	HP Easy Printer Care CVE: 2011-4786	MS
BitDefender Online Scanner CVE: 2007-6189	HTML	Honeywell HSC Remote Deployer CVE: 2013-0108	MS
GlobalLink 2.7.0.8 CVE: 2007-5722	HTML	iMesh 7 ‘IMWebControl’ CVE: 2007-6493	HTML
D-Link Audio Control CVE: 2008-4771	HTML	KingSoft ‘UpdateOcs2.dll’ CVE: 2008-1307	HTML
Lycos FileUploader CVE: 2008-0443	HTML	HP Compaq Notebooks CVE: 2007-6333	HTML
Control Webpage	HTML		

APPENDIX B. SCRAPY AND MONGODB SCRIPTS

This appendix contains Python 3 code we wrote for configuring the Scrapy Web crawler, and Python 3 code for interfacing with MongoDB and extracting collected file samples. Scrapy installation and environment configuration documentation can be found at (Scrapy.org, 2022). The settings.py program includes the changes made to the WebsiteSpider class used in our environment.

web_spider.py

```
import scrapy
from faker import Faker
from tqdm import tqdm
from datetime import datetime
faker = Faker()

def ts():
    return "{:%Y-%m-%d %H:%M:%S}".format(datetime.now())
class WebsiteSpider(scrapy.Spider):
    name = "websites"

    def start_requests(self):
        global found_urls
        print("Generating random URLs")
        urls = ["NULL"] * 6000000
        for index, url in (enumerate(tqdm(urls))):
            urls[index] = ("http://" + faker.ipv4())
        print("URL generation complete\n")
        textfile = open("./logs/generated_ip_" + ts() +".txt","w")
        for element in urls:
            textfile.write(element + "\n")
        textfile.close()
        found_urls = []
        tf = open("./logs/found_urls_" + ts() + ".txt", "w")
        print("Begining web crawl")
        for url in tqdm(urls):
            try:
                yield scrapy.Request(url=url, callback=self.parse)
            except:
                continue
        i = 0
        for element in found_urls:
            tf.write(element + "\n")
            i += 1
        tf.close
        print("Web crawl complete")

    def parse(self, response):
        global found_urls
        found_urls.append(response.request.url)
```

settings.py

```
ROBOTSTXT_OBEY = True

SCHEDULER_PRIORITY_QUEUE =
'scrapy.pqueues.DownloaderAwarePriorityQueue'
CONCURRENT_REQUESTS = 5000
REACTOR_THREADPOOL_MAXSIZE = 5000
LOG_LEVEL = 'INFO'
COOKIES_ENABLED = False
RETRY_ENABLED = False
DOWNLOAD_TIMEOUT = 15
REDIRECT_ENABLED = True
AJAXCRAWL_ENABLED = True
DEPTH_PRIORITY = 1
SCHEDULER_DISK_QUEUE = 'scrapy.squeues.PickleFifoDiskQueue'
SCHEDULER_MEMORY_QUEUE = 'scrapy.squeues.FifoMemoryQueue'
COOKIES_ENABLED = False
```

To interface with MongoDB and extract the collected file samples, we used the Python package PyMongo. Installation and configuration instructions can be found at (Mongo.org, 2022).

mongofiles.py

```
from pymongo import MongoClient
import gridfs
import os
from tqdm import tqdm

client = MongoClient(host='localhost', port=27017)
db = client["thug.fs"]
counter = 1
fs = gridfs.GridFS(db)
print("connected to mongodb")

for document in tqdm(db.fs.files.find()):
    my_id = document['_id']
    outputdata = fs.get(my_id).read()
    if os.path.isfile("/home/<user>/<folder>/" + str(my_id)):
        break
    else:
        download_location = "/home/<user>/<folder>/" + str(my_id)
        output = open(download_location, "wb")
        output.write(outputdata)
        output.close()
```

LIST OF REFERENCES

Caviglione, L., Choraś, M., Corona, I., Janicki, A., Mazurczyk, W., Pawlicki, M., & Wasielewska, K. (2021). Tight arms race: Overview of current malware threats and trends in their detection. *IEEE Access*, *9*, 5371–5396. https://doi.org/10.1109/access.2020.3048319

Cisco Systems, Inc. (2022). *ClamAV documentation. Signatures – ClamAV documentation*. Retrieved November 21, 2022, from https://docs.clamav.net/manual/Signatures.html

Cisco Systems. (2022). *ClamAV*. http://www.clamav.net/

Conner, B. (2022). *Mid-Year Update: 2022 SonicWall cyber threat report*. https://www.sonicwall.com/medialibrary/en/infographic/mid-year-update-2022-sonicwall-cyber-threat-report.pdf

Dell'Aera, A. (2022). *Thug documentation*. https://buffer.github.io/thug/doc/

Dell'Aera, A., Guida, G., & Skovoroda, A. (2022). *Libemu* [C]. https://github.com/buffer/libemu (Original work published 2015).

Dell'Aera, A., Schloesser, M., & Piccinno, F. (2022). *PyLibemu* [Python]. https://github.com/buffer/pylibemu (Original work published 2011).

Dell'Aera, A., & Syme, P. (2022). *STPyV8* [C++]. *Cloudflare*. https://github.com/cloudflare/stpyv8 (Original work published 2019).

Eicar.org. (2022). *EICAR standard anti-virus test file*. https://www.eicar.org/download-anti-malware-testfile/

Fuentebella, C. (2022, November 11). *Trojan.HTML.IFRAME.FASGU – Threat Encyclopedia*. Threat Encyclopedia. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/trojan.html.iframe.fasgu/

Google.com. (2022). *GoogleBot*. https://developers.google.com/search/docs/crawling-indexing/googlebot

Google.com. (2022). *V8 JavaScript engine* [C++]. https://v8.dev/

Hjelmvik, E. (2011). *Passive OS fingerprinting*. Netresec. https://www.netresec.com/?page=Blog&month=2011-11&post=Passive-OS-Fingerprinting

Hjelmvik, E. (2022). *NetworkMiner (2.7.3) [C#]*. NETRESEC. https://www.netresec.com/?page=NetworkMiner

Hjelmvik, E. (2022). *NetworkMiner—The NSM and network forensics analysis tool*. Netresec. https://www.netresec.com/?page=NetworkMiner

Honeynet.org. (2022). *The Honeynet project*. https://www.honeynet.org/about/

Ikinci, A. (2008). *The Monkey-Spider project*. https://monkeyspider.sourceforge.net/documentation.html

Ikinci, A., Holz, T., & Freiling, F. (2008). *Monkey-spider: Detecting malicious websites with low-interaction honeyclients*. https://madoc.bib.uni-mannheim.de/27368/

Invernizzi, L., Comparetti, P. M., Benvenuti, S., Kruegel, C., Cova, M., & Vigna, G. (2012). EvilSeed: A guided approach to finding malicious web pages. *2012 IEEE Symposium on Security and Privacy*, 428–442. https://doi.org/10.1109/SP.2012.33

Joshi, R. C., & Sardana, A. (2011). *Honeypot: A new paradigm to information security*. CRC Press.

Keats, S., Nunes, D., & Greve, P. (2007). Mapping the mal web. *McAfee SiteAdvisor*, 25.

Lake, J. (2019, December 13). *What is a Drive-by Download and how can it infect your computer?* Comparitech. https://www.comparitech.com/blog/information-security/drive-by-download/

Le, V. L., Welch, I., Gao, X., & Komisarczuk, P. (2013). Anatomy of drive-by download attack. *Proceedings of the Eleventh Australasian Information Security Conference*, *138*, 49–58.

Lookyloo.eu. (2022). *Looklyloo*. https://www.lookyloo.eu/docs/main/index.html

Microsoft. (2017). Exploit:HTML/Phominer.A. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Exploit%3AHTML%2FPhominer.A

Metasploit.org. (2022). *Rapid 7*. https://github.com/rapid7/metasploit-framework

Mongo.org. (2022). *MongoDB database 6.1.0*. https://github.com/mongodb/mongo

Mongo.org. (2022). *PyMongo 4.3.3 documentation*. https://pymongo.readthedocs.io/en/stable/ [Documentation].

Namanya, A. P., Cullen, A., Awan, I. U., & Disso, J. P. (2018). The world of malware: An overview. *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, 420–427. https://doi.org/10.1109/FiCloud.2018.00067

Offensive Security.org. (2022). *Exploit database archive*. https://www.exploit-db.com/

Pinoy, J., Van Den Broek, F., & Jonker, H. (2021). *Nothing to see here!* [Open University of the Netherlands]. https://research.ou.nl/

Qassrawi, M. T., & Zhang, H. (2011). Detecting malicious web servers with honeyclients. *Journal of Networks*, *6*(1), 145–152. https://doi.org/10.4304/jnw.6.1.145-152.

Rowe, N. C., & Rrushi, J. (2016). *Introduction to cyberdeception*. Springer International Publishing. https://doi.org/10.1007/978-3-319-41187-3

Scrapy.org (2022). [Python]. *Scrapy project*. https://github.com/scrapy/scrapy (Original work published 2010).

Seifert, C. (2006). *HoneyC* (1.3.0). *The Honeynet Project*. https://www.honeynet.org/projects/old/honeyc/

Seifert, C., & Steenson, R. (2006). *Capture-HPC*. The Honeynet Project. https://www.honeynet.org/projects/old/capture-hpc/

Stewart, M., Ohlinger, M., Czechowski, A., & Greg, L. (2022). *Windows 11 overview*. https://learn.microsoft.com/en-us/windows/whats-new/windows-11-overview

Stichting Cuckoo Foundation. (2019). *Cuckoo sandbox—Automated malware analysis*. https://cuckoosandbox.org/

Twisted Matrix Labs. (2022). [Python]. *Twisted*. https://github.com/twisted/twisted (Original work published 2011).

VirusTotal.org. (2022). [Go]. *VirusTotal CLI*. https://github.com/VirusTotal/vt-cli (Original work published 2018).

THIS PAGE INTENTIONALLY LEFT BLANK