Hewlett Packard Enterprise

Trust, but verify: Evaluating DNS-based malware detectors

Technical white paper

Contents

Introduction

While enterprise networks have tightened the screws on various ingress and egress channels, Domain Name System (DNS) traffic has always enjoyed a sense of partiality. This may partly be due to its limited, yet essential functionality. DNS[1] represents the core infrastructure that is used to query for the IP address of a given domain. With this service, the user needs to remember one static URL while the underlying IP address may be dynamically updated or changed. While DNS improves the usability of the Internet for human users, the same can be said for malware as well. Malware typically uses DNS to identify and establish communication with its command and control (C&C) center.

This paper explores the ways in which malware uses DNS, how today's malware detectors attempt to solve this problem, and a method to evaluate the effectiveness of such malware detectors.

DNS and malware

Typically, malware exhibits three phases of communication—registration, C&C communication, and data exfiltration. Of these, most malware use DNS for two purposes—registration with its C&C and exfiltration of data.

C&C registration

Once a malware instance is installed on a victim,[2] it will try to establish a communication channel with its master, which is also known as the C&C (or C2) server. This process is labeled as registration. The server's address is generally not hard-coded in the malware to avoid detection by reverse engineers. Instead, techniques such as domain generation algorithms (DGAs)[3] are used to create pseudo-random domains that vary over time. When using such techniques, it is necessary for the malware to search for multiple domains before it identifies the C&C. This search is done by sending DNS queries to different domains until the right one responds.

This is typically the noisiest symptom of malware since it sends multiple requests over a short period to connect with its C&C.

Data exfiltration

Once C&C communication is established, the malware may execute different actions (for example, read the contents of its victim) and send results over to its master. Traditional malware is used to accomplish this task by communicating over HTTPS to mask its traffic. However, recent malware detectors have employed various techniques to crack down on such exfiltration methods. Hence, malware authors have started exfiltrating data over DNS. While the process is slower than traditional exfiltration, it can be stealthier, thus allowing malware to go undetected for a long time.

The malware creates a DNS query targeting the C&C's domain to exfiltrate data over DNS. While doing so, it encodes the data to be sent into a string and is attached as a subdomain to the query. Recursive DNS servers ensure the delivery of this request to the C&C server when sending such a query and the server may decode the subdomain to retrieve the data snippet. Restrictions in the DNS protocol limit the amount of data to be transferred this way hence the malware may need to send multiple DNS queries to exfiltrate a single piece of data.

DNS malware analytics

Recently, malware detectors have taken notice of this and added DNS feeds into their scope of inspection. The detectors have attempted to automate the detection of malicious behavior over that vector as well.

As discussed earlier, malware tends to generate noisy DNS traffic during the registration phase. Due to this fact, most malware detectors concentrate their detection algorithms here. Also, the newer malware uses DGAs that can generate human-readable domains, thus making it harder for automated algorithms to identify them. The large volume of DNS registration traffic and the complexity of isolating the malware from this data have yielded several statistical and machine learning models designed to detect malware. These models typically try to identify malware based on the analysis of domain names such as the frequency of characters, ill-formed words, and more. Or, with the analysis of traffic patterns such as the number of requests over a short period, consistent bursts of requests over a given period, and more.

[1] computer.howstuffworks.com/dns.htm
[2] A computer infected by malware
[3] resources.infosecinstitute.com/domain-generation-algorithm-dga/

Validating an analytics module

While malware detectors continue to implement “cutting-edge” algorithms to detect complex malware, there is not a reliable way to validate the claims of such detectors. When a new malware variant is detected in the wild, security professionals are often left to guess how their security deployments would react. Today’s best-case scenario involves intentionally infecting a test environment to replicate the behavior of the malware detector in the production environment. However, the risk of a wider infection and the inability to execute a malware on-demand may trigger the incorrect behavior, thus leading to false positives.

Maltese[4] (malware traffic emulating software) was designed to overcome this exact scenario. By emulating a given malware on-demand, security professionals can verify the detection capabilities of their security deployments without risking a real infection. The following sections describe a sample environment to use Maltese, along with its use cases.

Emulating DNS traffic

When creating a malicious traffic pattern, there are primarily two goals to consider—the generation of domains and the traffic pattern based on which queries are sent over the network.

Maltese is written in Python and uses Scapy[5] to create its DNS queries. The tool is implemented based on a plugin architecture, thus allowing it to be easily extended for newer malware variants. There are two primary ways of emulating DNS traffic—replaying existing artifacts and using a domain generation algorithm to create new domains.

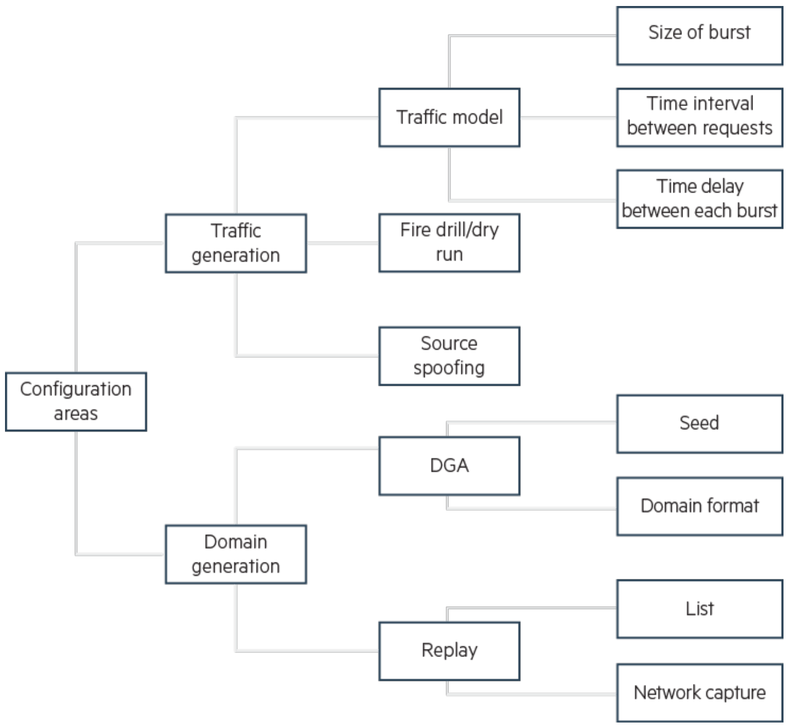

Figure 1. Configuration areas to consider while designing DNS traffic

[4] github.com/hpe-appliedSecurityresearch/maltese
[5] secdev.org/projects/scapy/

Replaying traffic

Security professionals tend to have a list of malicious domain indicators or network captures with malicious activity. These artifacts can be replayed as necessary to recreate a malware's presence in the environment.

Domain generation algorithms

Newer malware variants use DGAs to avoid their domains from being blacklisted easily. A DGA can generate pseudo-random domains based on a given seed. Typically, malware tends to use the current date as the seed. Hence, the set of domains that are contacted varies each day making domain blacklisting very difficult.

Malware analysts tend to publish the DGAs once they reverse engineer the samples. Maltese allows the implementation of such DGAs to generate domains and send DNS requests accordingly.

Traffic model

While the replay and DGA plugins provide the domains to be sent, the traffic model defines the pattern in which these requests are sent. This model is designed based on the assumption that malware tends to communicate with its C&C in bursts of traffic.

Based on this assumption, the traffic model considers three primary variables.

The size of a burst

This defines the number of DNS queries sent in a burst. The model may be designed to make this number random or within a range, such that a subtle variance may be introduced.

The time delay between bursts

Some malware may send out a burst of queries once a day. Others may do so once an hour or pick random delays. This parameter allows such variations to be configured to all malware detectors to track consistent bursts. The parameter may be tweaked to establish the variations that the detector can and cannot detect, thus allowing for a better understanding of its capabilities.

The time interval between requests within a burst

While some malware may send as many queries as fast as possible, other malware instances may be stealthier and increase the time delay between requests in a burst. Such behaviors may be configured using the interval parameter.

These parameters may be tweaked to introduce pseudo-random time and burst variances that closely emulate the behavior of real malware samples.

The environment

While setting up Maltese, there are two essential components—the client and the server. The client would be a victim's machine that would be potentially infected by malware. This is where Maltese would be installed to emulate the presence of real malware. The server is a DNS server that would respond to the malware's DNS queries. There are two potential ways to set up Maltese—as part of the live environment or in an isolated environment.

Deploying in a live environment

The advantage of using a live environment is that the infrastructure is already in place. Also, this would evaluate the malware detectors in their actual deployment, rather than a test setup. Hence, it would be easier to understand the reactions of a security deployment in their original habitats.

While using such a setup, Maltese would be installed on a "victim" machine and triggered to emulate a given malware. The emulated traffic would be tapped by the malware detector along with benign traffic from the rest of the network. Based on the time and type of event triggered in Maltese, a corresponding detection is expected from the malware detector. Lack of a valid detection would indicate a false negative in the live environment.

Note

By default, Maltese sends its DNS queries to the DNS server configured on the victim's machine.

Deploying in an isolated environment

Sometimes, it may not be an option to run the validation process in a live environment. In such cases, an isolated environment may be set up to look like one.

While validating a DNS-based malware detector, there are three essential entities involved—the malware that is being tested, a DNS server, and the malware detector under study. In this case, Maltese would be used in place of the real malware. An isolated environment may be set up with Maltese on one machine, the DNS server on another machine, and the malware detector on a third machine. The environment is set up such that Maltese and the DNS server can communicate to each other, and that the malware detector can tap on the traffic between them.

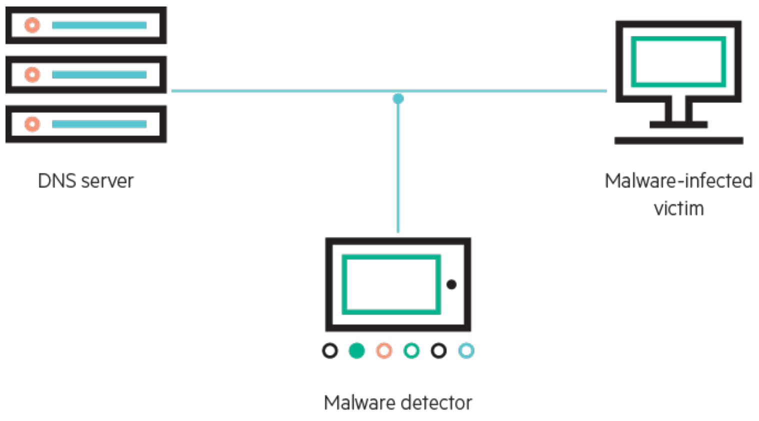

Figure 2. Example of an isolated environment

Note

While a real malware sample could be safely used in an isolated environment, it is still a challenge to execute malware on-demand.

Once the setup is ready, Maltese will be triggered to emulate a given type of malware, and the malware detector would be observed for any alerts. The lack of an alert would indicate a false negative, while the presence of an alert indicates a true positive.

This setup may be used to emulate the presence of a specific malware in a given host. It is necessary to emulate an entire network to recreate a realistic scenario, with one or few infected machines. This may be achieved in two different ways.

Replaying existing network captures

In this method, DNS traffic of a live network is captured and replayed in the isolated environment. The malicious traffic from Maltese is spoofed to one of the existing IP addresses from the capture, thus emulating the presence of a specific malware in the network. In this method, the existing capture might have some unknown malicious traffic as well. Hence, it is possible for the malware detector to find some unexpected malicious patterns during the process. This noise should be accounted for while verifying the results of the malware detector.

Creating new network traffic

It is recommended to run multiple instances of Maltese to emulate an entire network from scratch. One instance of Maltese represents one instance of a malicious or benign host. Hence, multiple instances of Maltese may be executed simultaneously to represent different hosts. The replay module can be used to send benign traffic instead of malicious captures to emulate the non-malicious hosts. Each instance of Maltese may be executed in a separate virtual machine or on a single machine but should be configured to spoof different source IP addresses.

Emulating a DNS server

In all these environments, Maltese represents only the malware instance, which is a client. All malware instances communicate with the C2 via the DNS server. By default, the DNS server configured on the host machine is used. But, if the need arises to construct the DNS responses as well, a separate DNS server may be set up within the environment and all machines involved in this setup can be configured to access this instance. Emulating a DNS server is typically necessary when malware detectors consider a successful DNS response as part of their detection algorithm. Otherwise, using the default DNS server would most likely generate NXDOMAIN responses for all the DNS queries.

Conclusion

Enterprises may choose different ways to verify their security deployments. This paper provides one such way. Nevertheless, the most important takeaway is that off-the-shelf malware detectors may be very effective in doing their job, but it is necessary to validate their capabilities to ensure that they suit the needs of the target enterprise. Trust, but verify.

Learn more at

hpe.com/software/hpsr

Sign up for updates

Hewlett Packard Enterprise

4AA6-7561ENW, September 2016