Normalizing Empire’s Traffic to Evade Anomaly-based IDS

Utku Sen
utku@utkusen.com

Gozde Sinturk
gozdesinturk@gmail.com

Abstract

Perimeter defenses are holding an important role in computer security. However, when we check the method of APT groups, a single spear-phishing usually enough to gain a foothold on the network. Therefore, red teams are mostly focused on ”assume breach” type of scenarios. In these scenarios, testers need to use a post-exploitation framework. Besides that, testers also need to hide the server-agent communication from NIDS (Network Intrusion Detection Systems). In this paper, we will discuss one of the most famous post-exploitation tool, Empire’s situation against payload-based anomaly detection systems. We will explain how to normalize Empire’s traffic with polymorphic blending attack (PBA) method. We will also cover our tool, ”firstorder” which is designed to evade anomaly-based detection systems. firstorder tool takes a traffic capture file of the network, tries to identify normal profile and configures Empire’s listener in such way.

1 Introduction

Since the beginning of computer security history, perimeter defenses are playing an important role. As we can observe from recent breaches, perimeter defenses are not holding attackers out from an organization’s network. A good crafted spear-phishing e-mail usually enough to gain a foothold on the network. More and more attackers are using these techniques to bypass these defenses. As attacker profile has been changed by time, traditional defense approaches are changing too. Today, organizations and red teams are mostly focused on ”assume breach” approach. Assume breach is simply accepting that attackers are bypassed your perimeter defenses and got a foothold on your network. So, an organization should focus on defense strategies as per that way. Also, red teams will be focused on post-exploitation instead of passing perimeter defenses.

We can show NIDS (Network Intrusion Detection Systems) as an initial step for detecting attackers on the network.[1] There are two main types of NIDS: signature-based and anomaly based. Signature-based NIDS refers to the detection of attacks by looking for pre-defined patterns of previously known attacks.[2] Anomaly-based NIDS refers to building a statistical model describing the normal network traffic, and flagging the abnormal traffic. Anomaly-based systems have the advantage of detecting zero-day attacks, since these attacks will be distinguishable from normal traffic. On the other hand, anomaly-based systems require a training phase in order to identify normal network traffic.[3]

It is possible to mislead learning algorithm of an anomaly-based system by poisoning the initial data.[4] However in a real-world scenario, it’s hard for an attacker to know, when the network is being trained for anomaly detection purposes. In addition, we do not see this attack type in major breaches by APT groups.[5]

In this paper, we will focus on post-exploitation framework Empire’s [6] situation against a payload based, application layer anomaly detector[3] after the normal traffic classification is already implemented.

2 Empire Project

Empire is a PowerShell and Python based post-exploitation framework which is designed for ”assume breach” type of activities.[6][8] We can describe Empire’s workflow in two parts: Agent and listener. Agent states the infected machine on the network which takes and executes given tasks on there.[6] Listener is described as a communication server (C2) in which agent connects there, gets designated task and sends related output of the task.[6]

It’s known that attackers are using Empire widely for malicious purposes.[19]

2.1 HTTP Listener

Empire supports HTTP and HTTPS listeners for C2-agent communication. Even tough HTTPS connection encrypts all communication, we will assume there is a solution on the network which intercepts and decrypts SSL/TLS traffic. Because of that, HTTP listener will be the main topic for this paper.

HTTP listener has following key options:

- KillDate: Date for the listener to exit
- DefaultDelay: Agent delay/reach back interval
- WorkingHours: Hours for the agent to operate
- DefaultProfile: User-agent value and URI specificion for the agent
- DefaultJitter: Jitter in agent reachback interval
- Port: Listening port of the C2 server
- StagingKey: Staging key for initial agent negotiation
- ServerVersion: Server header for the C2 server.

2.2 C2-Agent Communication and Packet Structure

HTTP listener provides encrypted communication even without SSL/TLS connection. This encryption is done in two parts:[9]

1) ”Client Data” is symmetrically encrypted with AES algorithm where encryption key is client’s session key. Packet structure looks like following chart:[9]

Table 1: AES(client data)

AES IV	Enc Packet Data	HMAC
16	% 16 bytes	10

2) ”Metadata/Routing Data” is symmetrically encrypted with RC4 algorithm where encryption key is ”StagingKey” of the listener. Because of that, all agents in the network can decrypt the ”Metadata/Routing Data” packet in order to route packets to right destination. Packet structure looks like following chart:[9]

Table 2: Routing Packet

RC4 IV	RC4s(RoutingData)	AES(client packet)
4	16	RC4 length

Table 3: RC4(RoutingData)

SessionID	Lang	Meta	Extra	Length
8	1	1	2	4

We will consider these encrypted payloads are not decryptable by a solution on the network since there is no publicly known product has Man-in-the-middle capability for Empire’s communication.

2.3 NIDS on Empire’s HTTP Listener

Assuming that the HTTP Listener and agent are located on the same local network. After the initial negotiation, agent will connect to C2 every n seconds which is defined in ”DefaultDelay” option of the listener. Here is a generic request and response of this heartbeat connection:

```
GET /read.php HTTP/1.1
Cookie: session=VAGyTO1KBPOOBxJ45BZrcm3BinQ
User-Agent: Mozilla/5.0 (Windows NT 6.1)
    AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/41.0.2228.0 Safari/537.36
Host: 192.168.1.26
Connection: close
```

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 208
Cache-Control: no-cache, no-store, must-
    revalidate
Pragma: no-cache
Expires: 0
Server: Microsoft-IIS/7.5
Date: Thu, 08 Feb 2018 17:57:40 GMT

<html><body><h1>It works!</h1><p>This is
    the default web page for this server.</
    p><p>The web server software is running
     but no content has been added.</p></
    body></html>
```

We can list following traits which can be insight of an anomaly-based NIDS:

1) **Request URI**: As shown in the request above, agent makes it’s connection to the C2 server with a GET request to a specific URI (”read.php” in the example) If only html or aspx pages are in use on the local network, this ”php” extension may flagged by the anomaly detection system.

2) **User-agent value**: User-agent value defines the operating system and browser choice of the agent. For example, if all users on the network uses Microsoft Windows with Chrome, setting user-agent value to macOS

with Safari may flagged by the anomaly detection system.

3) **Server header**: Server header value defines the web server type of the C2. For example, if all the servers on the network are using Linux, setting server header as ”Microsoft-IIS” may flagged by the anomaly detection system.

4) **Default HTML Content**: This is related with server header. If only Linux systems are used on the network, IIS default page may flagged by the anomaly detection system.

5) **Port**: If only common ports like 80, 443, 8080 are used on the network, selecting communication port as 5839 may flagged by the anomaly detection system.

6) **Connection Interval (DefaultDelay)**: By default, agent will send heartbeat request to the C2 server in every 5 seconds. If regular users are not connecting to a local server in every 5 seconds, this will be likely to flagged by anomaly detection system.

7) **POST Request Body**: As seen in the request above, POST request body contains non-printable characters. If all users are browsing regular websites, this will be likely to flagged by anomaly detection system.

We can gather the traits explained above in two different groups: Traits that can be changed in Listener's option menu, traits that can be changed by changing Empire's source code. Our tool, ”firstorder” won't cover the second group. However, possible solutions will be discussed in this paper.

First group consists following traits: Request URI, User-agent value, Server header, Port, Connection interval

Second group consists following traits: Default HTML Content, POST Request Body

3 Proposed Solution

Polymorphic blending attack (PBA) is a useful technique to evade anomaly-based systems.[10][11] The idea behind the PBA is creating attack packets which are matches to normal traffic profile.[10] To execute this technique, the attacker should know the features which are used to train normal traffic profile. In order to achieve that, our model requires a traffic capture data of the network after the normal traffic training is done. By checking the traffic capture data, our model will know what kind of traffic is used frequently. As a result, our main attack steps will be:

1. Get traffic capture data of a normal traffic and define normal behaviour of users.
2. Change Empire's listener traits according to first step.
3. Start the C2-agent communication.

To normalize the first group of traits (explained in the previous section), we will extract most common URI, user-agent, server header and port values from the traffic capture data. With these data, we will set appropriate listener options.

However, finding normal connection interval value is not an easy task. One way to do that is figuring the connection interval and frequency of users to the specific web sites. However, this solution will not be practical since there can be delays, interruption during traffic capture and this will affect the recorded connection interval of users.

The second solution relies on false-positive rates of anomaly detection systems. The false-positive rate of an anomaly detection system has a positive correlation with the size of the network [12] Which means that in a large network, even if we keep connection interval value small and get flagged by anomaly detection system, this will be most likely to be seen as false-positive by analysts. However, if we are located in a small network, we need to set connection interval higher than a pre-defined threshold. The disadvantage of this method is C2-agent communication will be delayed. It is a trade-off to keep our communication out of sight from the anomaly detection system.

For the second group of traits (explained on previous section) default HTML content can be chosen by identifying most visited websites in traffic capture data. However, normalizing POST request body of the communication is not achievable by using traffic capture data. As it explained in previous sections, POST requests are encrypted and contains non-printable characters. If we don't use encryption in there, signature-based NIDS probably will flag them as malicious since they may contain specific keywords like ”whoami”.

Instead of encrypting the data, we can use ”Markov Obfuscation” method[13] As shown by Cylance SPEAR Team, a model which is trained by an English book, looks like a normal HTTP traffic. It's hard to differentiate it from a blog posting traffic. To apply this to C2-agent communication, we will use MarkovObfuscation tool[17] In order to do that, we need both change Empire's agent packet handler code and the code inside the stager which generates agent packets. However, we will not change encryption mechanism since it may break lots of functionalities. We will apply Markov Obfuscation to the encrypted packet. Operation steps will be (from agent-side perspective):

1. Encode encrypted data with Base64 to get rid of non-printable characters
2. Train encoded data with MarkovObfuscation's book dataset[18]

3. Send Markov encoded data to C2 server

Assuming the ”data.txt” contains base64 encoded encrypted data, the command will be:

```
python obfuscate.py -f book datasets/98.txt
     data.txt
```

Which will output:

```
from the prisoner evremonde . the door
    steps got up and the last night and now
     . the day . he bent over the . i shall
     be . the eyes . i came to the prisoner
     had had been a certain . the doctor
    said the . a . the wall . the prisoner
    of her . the first . and was a . be .
    and stood . the door was in the door
    threw out again . the face . i hope .
    the . of the . by the door and then . i
     don't know that the two . the time . i
     shall be . he sat down . he did not .
    and to the door was to the doctor said
    the . madame defarge . the people . the
     window . the prisoner of his . the
    other . he knows . the . we have seen
    by herself . the prisoner there i am
    not so . he did not . i do you . which
    he had in the prisoner was so . i had
    been . it was not being . i was a .
    miss pross . he and . and she had .
    defarge . the prisoner . the . the . i
    do you know well worn out of the little
     . i remember
```

When we look at from C2-side, operating steps will be:

1. Decode Markov obfuscated data with same train set
2. Decode Base64 encoded data
3. Decrypt encrypted content

Assuming the ”data.txt” contains Markov obfuscated data, the command will be:

```
python obfuscate.py -d -f book datasets/98.
    txt data.txt
```

Which will output base64 encoded encrypted data

To use this method, you have to train data on both agent and server side. This will be time consuming and agent will use lots of resources of the infected machine. On the other hand, you need to drop train data to the infected computer which will damage the OPSEC needs. When we consider these disadvantages, we can state that this method will not be practical for real-world scenarios.

4 firstorder Tool

firstorder tool is written in Python[14] mostly based on scapy library[15]. With given pcap file[16] it can extract following details of the network:

1. Most used ports
2. Most used server headers
3. Most used user-agents
4. Most used URIs
5. How many different machines broadcasted ARP packets (for determining network size)
6. How many different machines executed LDAP queries (for determining network size)

According to first four gathered information, firstorder will set appropriate options of Empire's listener via Empire's API. With the last two information, firstorder calculate number of computers on the network. According to number of computers, connection interval values will be:

Table 4:

Number of Computers	Connection Interval
25 or less	25 seconds
25-50	20 seconds
50-75	15 seconds
75-100	10 seconds
100 or more	5 seconds

4.0.1 Usage

firstorder has two different modes: In the first mode, it only analysis the capture data but doesn't start an Empire listener. Example command:

```
python firstorder.py capture.pcap
```

In the second mode, it connects to Empire's API and creates a listener according to analyze result. Assuming that user has started Empire in REST API mode with following command:

```
python empire --rest --username empireadmin
     --password Password123
```

firstorder's command will be:

```
python firstorder.py capture.pcap --
    username empireadmin --password
    Password123
```

Our test data is captured from a private network which allows us to use the data on our tests. File properties are:

- File Size: 575MB
- Captured Packets: 715639
- Capture Time: 2380 seconds

When we analyze it with firstorder, job is finished in 7 minutes 28 seconds. As a result, firstorder detected following statistics:

1. 352 different IP addresses broadcasted ARP packet
2. 303 different computers used LDAP protocol
3. Most used user-agent is: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36
4. Most used URIs: /welcome.aspx, /1515920640761/redot.js, /sporarena/galeri-arda-turan-survivorda-herkes-bunu-konusuyor-40738732
5. Most used server header: Microsoft-IIS/8.0
6. Mosed used Port: 1104 (Proxy Port)

5 Conclusion

Anomaly-based detection systems fills the gap which signature-based systems cannot. These systems are helpful against zero-day attacks. However, like all the defense systems, they have potential weaknesses. Since they trust the traffic which matches with the normal profile, if an attacker can analyze and describe the normal traffic, he/she can configure Empire's communication mechanism to evade anomaly-based systems. As a result, anomaly-based systems should not be seen as a silver bullet for network security.

References

[1] Hazem M. El-Bakry, Nikos Mastorakisa, Real-Time Intrusion Detection Algorithm for Network Security,WSEAS Transactions on communications, Issue 12, Volume 7, December 2008

[2] Brandon Lokesak (December 4, 2008). "A Comparison Between Signature Based and Anomaly Based Intrusion Detection Systems" (PPT). www.iup.edu.

[3] Jyothsna, V., Prasad, V. V. (2011). A Review of Anomaly based IntrusionDetection Systems. International Journal of Computer Applications, 28(7).

[4] Biggio, B., Corona, I., Nelson, B., Rubinstein, B., Maiorca, D., Fumera, G., Giacinto, G. (n.d.). Security Evaluation of Support Vector Machines in Adversarial Environments.

[5] Indicators of Attack vs. Indicators of Compromise . (2017, May 12). Retrieved from https://www.crowdstrike.com/blog/indicators-attack-vs-indicators-compromise/

[6] EmpireProject/Empire. (2018, February 7). Retrieved from https://github.com/EmpireProject/Empire

[7] Wang, K., Stolfo, S. (n.d.). Anomalous Payload-based Network Intrusion Detection.

[8] Harmj0y, Jaredcatkinson. (n.d.). Empire Your best Friend to Secure Your Systems. Retrieved from https://hackcon.org/uploads/326/04

[9] EmpireProject/Empire. (n.d.). Retrieved from https://github.com/EmpireProject/Empire/blob/master/lib/common/packets.py

[10] Fogla, P., Lee, W. (n.d.). Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques.

[11] Gibbs, P. (2017). Intrusion Detection Evasion Techniques and Case Studies. SANS Institute.

[12] Narsingyani, D., Kale, O. (2015). Optimizing false positive in anomaly based intrusion detection using Genetic algorithm. 2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE). doi:10.1109/mite.2015.7375291

[13] Wolff, M., Wallace, B., Zhao, X. (2017). Applied Machine Learning For Data Exfil And Other Fun Topics. Retrieved from https://www.blackhat.com/docs/us-16/materials/us-16-Wolff-Applied-Machine-Learning-For-Data-Exfil-And-Other-Fun-Topics.pdf

[14] Welcome to Python.org. (n.d.). Retrieved from https://www.python.org/

[15] Scapy. (n.d.). Retrieved from http://www.secdev.org/projects/scapy/

[16] Development/LibpcapFileFormat - The Wireshark Wiki. (n.d.). Retrieved February 10, 2018, from https://wiki.wireshark.org/Development/LibpcapFileFormat

[17] CylanceSPEAR/MarkovObfuscate. (n.d.). Retrieved from https://github.com/CylanceSPEAR/MarkovObfuscate

[18] CylanceSPEAR/MarkovObfuscate. (n.d.). Retrieved from https://github.com/CylanceSPEAR/MarkovObfuscate/blob/master/datasets/98.txt

[19] Hunting The Known Unknowns (With PowerShell). (n.d.). Retrieved from https://conf.splunk.com/files/2016/slides/hunting-the-known-unknowns-the-powershell-edition.pdf