Malwares From Thin Bits

Itzik Kotler,
CTO & Co-Founder of SafeBreach

About Me

- 15+ years in InfoSec
- CTO & Co-Founder of SafeBreach
- Presented in DEF CON, Black Hat, RSA, HITB, THOTCON, CCC, ... But it’s my first time in BSidesDFW! I love it!
- http://www.ikotler.org

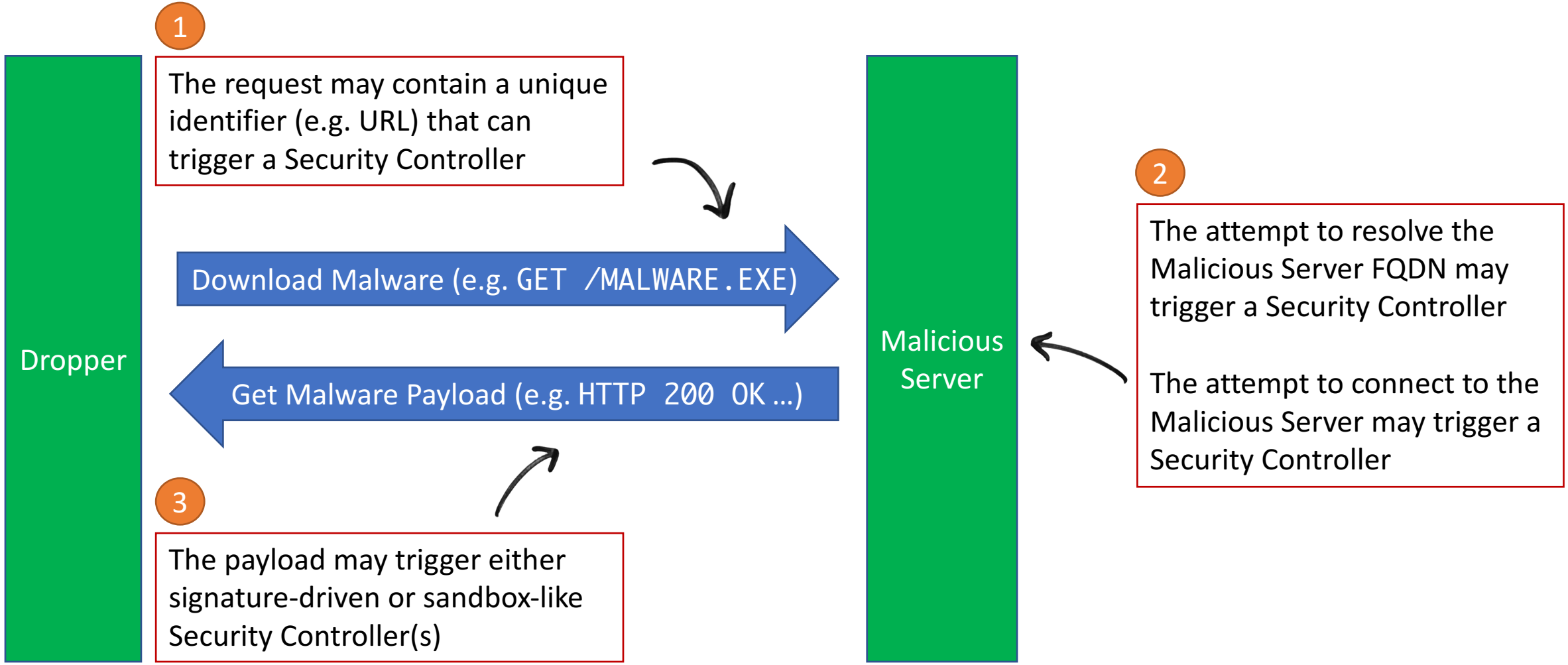
Drop It Like It’s Hot (From Wikipedia)

- A dropper is a program (malware component) that has been designed to "install" some sort of malware (virus, backdoor, etc.) to a target system.

- The malware code can be contained within the dropper (single-stage) in such a way as to avoid detection by virus scanners or **the dropper may download the malware to the target machine once activated (two stage)**

THE SUBJECT OF THIS TALK!

Common Failure Points in Two-Stage Droppers



The Server Dilemma

- We can normalize the Dropper request, such that it won't contain a unique identifier in it. (e.g. `GET /` vs `GET /mymalwaregen.php?id=5`)

- We can play with the Malicious Server payload to make it look more innocent (e.g. Steganography, Ad-hoc Compressing Algorithm etc.)

- We can't programmatically make our Malicious Server to be innocent. Rotating / Replacing it every **Nth** interval is not a real solution …

The Untouchable Server Solution

- What if the server the Dropper communicated with is untouchable (reputation-wise), in other words a popular website like Yahoo?
- That'd normally requires some kind of access to Yahoo to host our payload and even then, it's temporarily … (until they will find out)
- What if the Dropper would treat website returned response as "encoded data" and will "decode it" in a way that will result in a creation of the malicious program?

How This Work? *Input → Code → Output*

- Data does not necessarily has to be plain or “AS IS”; it can be encoded, compressed, encrypted etc.

- Code can take data and *transform* it, it can be one-way (e.g. `MD5`, `SHA256` etc.) or two-way (e.g. `Base64`, `Zip`, `AES` etc.) function.

- If the input data *A* is static, one can write a code that will *transform* it to the desired output data *B*. It’s like it was “encoded” in the original form and the code is “decoding” it in run-time …

Example: Hello “BSidesDFW” *from* Yahoo!

Python 2.7.13

- 1st Step: Fetch 9 bytes (i.e. len(“BSidesDFW”)) from Yahoo

```
>>> import urllib2
>>> raw_data = urllib2.urlopen('http://www.yahoo.com').read(9)
>>> raw_data
'<!DOCTYPE’
```

- 2nd Step: Normalize The Data (ASCII Values)

```
>>> norm_data = map(ord, raw_data)
>>> norm_data
[60, 33, 68, 79, 67, 84, 89, 80, 69]
```

Hello “BSidesDFW” *from* Yahoo! (Cont.)

- 3rd Step: Perform a Series of Transformations* on The ASCII Values

```
>>> norm_data[0] -= -6      # i.e. 60-(-6)=66  , chr(66) is equal to ‘B’
>>> norm_data[1] -= -50     # i.e. 33-(-50)=83 , chr(83) is equal to ‘S’
>>> norm_data[2] -= -37     # ...
>>> norm_data[3] -= -21
>>> norm_data[4] -= -34
>>> norm_data[5] -= -31
>>> norm_data[6] -= 21
>>> norm_data[7] -= 10
>>> norm_data[8] -= -18
```

** In this case, Transformations are basic arithmetic operations but it can be anything …*

Hello “BSidesDFW” *from* Yahoo! (Cont.)

- 4th Step: Convert ASCII Values to String

```
>>> final_data = ''.join(map(chr, norm_data))
>>> final_data
'BSidesDFW'
```

Meet `mkmalwarefrom`

- Version: 1.0 (Initial Release)
- Programming Language: Python
- License: 3-Clause BSD
- Git Repository: https://github.com/SafeBreach-Labs/mkmalwarefrom

[✓] Transform Innocent HTTP Responses to Malicious Payloads

[✓] Transform Innocent Files Content to Malicious Payloads

Static Transformation
+
Internet Connection

Method #1: It's All About The HTTP Response

- Given a popular website like Yahoo that anyone can interact with (i.e. does not require login)
- Sending a `HTTP GET REQUEST` to Yahoo will yield back a `HTTP 200 OK RESPONSE` with `HTML` etc., plenty of data to work with
- A code can *transform* the Yahoo `HTTP 200 RESPONSE OK` into a malicious program by preforming a series of bitwise operations on it

Demo!
(*Generating /bin/ls from Yahoo!)*

```
$ git clone https://github.com/SafeBreach-Labs/mkmalwarefrom
$ cd mkmalwarefrom
$ cat /bin/ls | ./mkmalwarefrom.py -1 http://www.yahoo.com > download_ls.py
$ python download_ls.py > ls2
$ md5 /bin/ls ls2
```

Pros / Cons of this Method

- (**PRO**) Input web page *A* can be used as Pre-Shared Key (PSK) as it has to be the **EXACT** same page every time (for content to be de-coded)
- (**PRO**) Pure Code; Can Be Easily Mutated / Recursively-Fed
- (**CON**) It will always produce the same output (i.e. given page *A* and code that transforms it, it will always generate output file *B*)
- (**CON**) It requires Internet Connection

Dynamic Transformation
+
Internet Connection

A Word About *Private* or *Anonymous* Reviews

- There are websites that will let you post and comment Anonymously by design. Why? Ideology, Embarrassed, Privacy Concerns etc.

- More at: http://www.localvisibilitysystem.com/2014/02/17/17-sites-that-allow-private-or-anonymous-reviews-of-local-businesses/ (*this is not the most up-to-date list, just an example of …)*

Method #2: Comment & Control

- Anonymously posted data can serve as C2. E.g.: Wikipedia, RateMDs etc.

- Posted data can be used a selector between *1…N* code transformation functions that are embedded in the Dropper. (**Input can still be static!**)

- Posted data can also be used as a “decoding” scheme that in turn can lead to endless number of transformations. (**Input can still be static!**)

Demo!
(Data as Selector)

```
$ git clone https://github.com/SafeBreach-Labs/mkmalwarefrom
$ cd mkmalwarefrom
$ python bsidesdfw_wiki_example.py
```

Pros / Cons of this Technique

- (**PRO**) Payloads are completely programmable and can be deferred to run-time
- (**PRO**) Pure Code; Can Be Easily Obfuscated / Mutated / Recursively-Fed
- (**CON**) Not too many websites lets you post anonymously data to ‘em
- (**CON**) It requires Internet Connection

Static Transformation
+
No Internet Connection (Offline)

Method #3: Someone Else's Data

- The Computer's (i.e. Dropped Endpoint) Filesystem can be used as an additional data source *to be transformed*

- Just like reading web page, we can open a file and read it's content. Like: *%SystemRoot%\System32\Shell32.dll* or */etc/passwd*

- This can be an alternative to when the Computer (i.e. Dropped Endpoint) have limited access to the Internet, or air-gapped

Demo!
(*Generating netcat from ssh)*

```
$ git clone https://github.com/SafeBreach-Labs/mkmalwarefrom
$ cd mkmalwarefrom
$ cat /usr/bin/nc | ./mkmalwarefrom.py -2 /usr/bin/ssh > mk_nc.py
$ python mk_nc.py > nc2
$ md5 /usr/bin/nc nc2
```

Pros / Cons of this Method

- (**PRO**) Input file *A* can be used as Pre-Shared Key (PSK) as it has to be the **EXACT** same file every time (for content to be de-coded)

- (**PRO**) Pure Code; Can Be Easily Mutated / Recursively-Fed

- (**CON**) It will always produce the same output (i.e. given file *A* and code that transforms it, it will always generate output file *B*)

Prior Art / References for this Method

- “Gauss contains a module named Godel that features an encrypted payload. The malware tries to decrypt this payload using several strings from the system and, upon success, executes it” ← *Similar Idea; Different Implementation*

- More at: https://securelist.com/the-mystery-of-the-encrypted-gauss-payload-5/33561/

Chain Your Methods!
(aka. Defense In Depth For Attackers)

Entry Point:

Method #3: Someone Else's Data

+

Method #1: It's All About The HTTP Response

+

Method #2: Comment & Control

Acknowledgement

- Thanks to Amit Klein (VP of Security Research at SafeBreach) for his help debating this idea with me ☺

Thank You!
Q&A

Email: itzik@safebreach.com
Twitter: @itzikkotler
GitHub: https://github.com/SafeBreach-Labs