SafeBreach

EVERY TIME YOU UPLOAD A MALWARE SAMPLE...

Online Sandboxing Services As a Data Exfiltration Intermediary

A SafeBreach Labs research by

Dor Azouri, Security Researcher, SafeBreach

March 2018

CONTENTS

SCOPE

Last July at Black Hat and DEFCON, Itzik Kotler and Amit Klein took us on the "**Adventures of A/V and the Leaky Sandbox**", and provided us with the supporting spacebin code. The technique demonstrated in "Adventures of A/V and the Leaky Sandbox" takes place in a highly restricted and segmented network, where they showed how one can make use of cloud AV solutions and sandbox services in order to exfiltrate data from such a nearly isolated network.

We chose to continue this research angle by taking advantage of the same techniques, but by demonstrating a way for an attacker to use online sandbox services as a means for exfiltration. This research addresses the same network circumstances, and resembles many of the required steps. However, this new research angle, i.e. using online sandbox services as a means for exfiltration, is only practical for attackers that have technical knowledge about their target network.

In the examples below, we demonstrate exfiltration using the Google Virus Total and www.hybrid-analysis.com services.

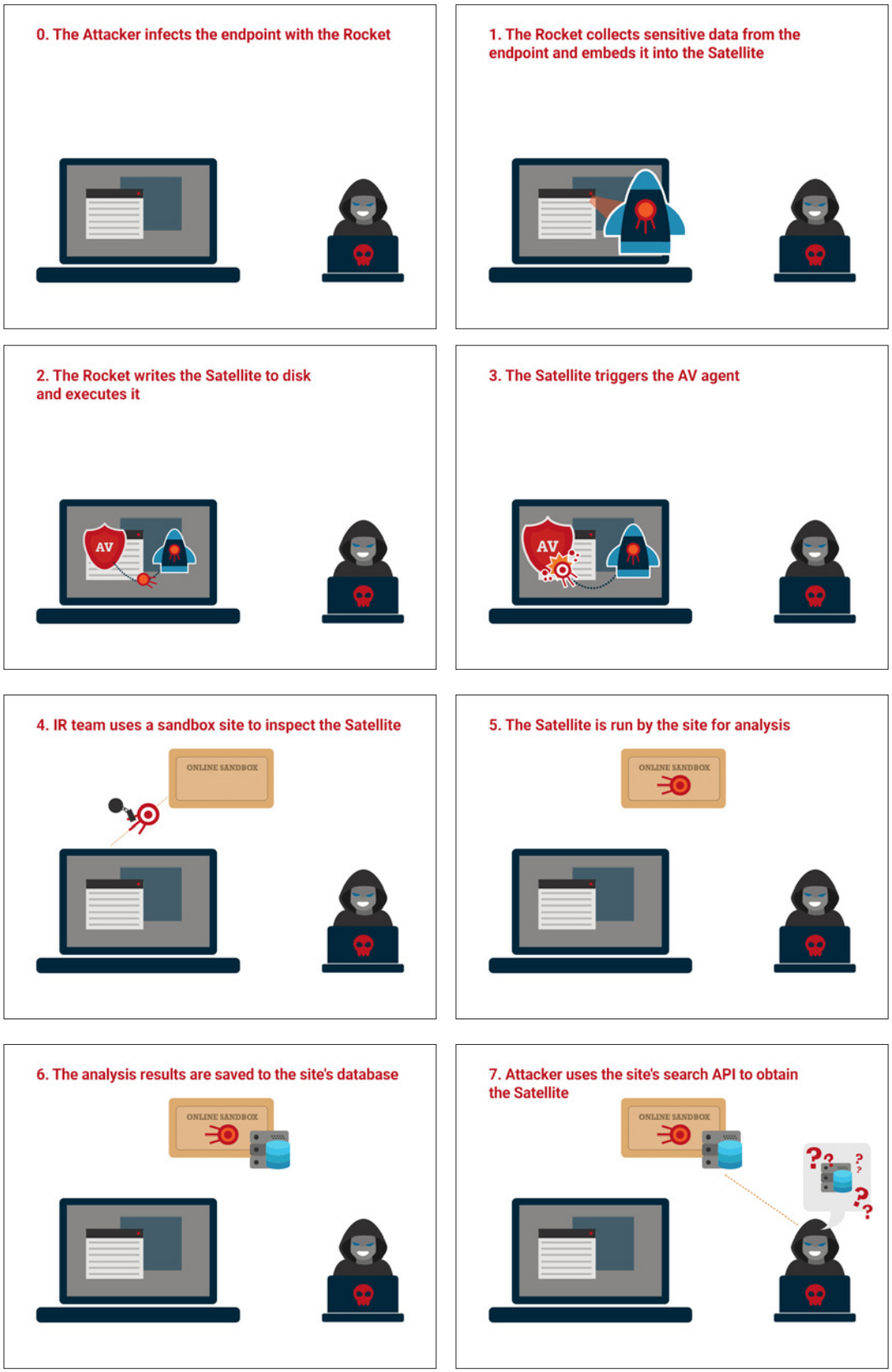
INTRO

For better comprehension, users are encouraged to read up on the original Kotler and Klein research - "The Adventures of AV and the Leaky Sandbox". Two crucial components are defined in the document: the rocket, and the satellite. In short, the rocket is the initial malware that generates the satellite from a predefined template. The satellite is a piece of code, initiated by the rocket to trigger an AV product.

One specific outcome that can result from triggering an AV product, is uploading the satellite to a sandbox service such as VirusTotal, Hybrid Analysis etc. In this post, we take advantage of this flow to exfiltrate the desired data, that was previously embedded into the satellite. Unlike the "Adventures of A/V and the Leaky Sandbox", we don't require the satellite to actively communicate out of the sandbox - instead, we use the sandbox service database itself as an intermediary for transferring data.

The exfiltration as a whole consists of two main objectives: incorporating the desired data into the satellite, and retrieving it by querying the sandbox service's databases. Both objectives can be achieved using various techniques. We will demonstrate some, and just list many other possible combinations.

The overall exfiltration scheme is illustrated in the following steps:



Let's first present the pros and cons that differentiate this method from the original method demonstrated by Kotler and Klein.

Pros:

- It relieves the satellite of the need to actively exfiltrate the data, meaning it no longer needs to emit outbound network traffic
- The attacker is less visible and harder to track down, as the gathering of the data is done passively from the sandbox service database (that the attacker queries)
- There is no need to run an HTTP server nor an authoritative DNS server

Cons:

- It can operate in narrower use cases - only in networks that practice a process of sending suspicious samples to an online sandbox engine
- Attacker needs prior specific information about his target network's policy which makes it only practical when he knows in advance which kind of sandbox service the organization is using
- The exfiltrated data remains public (though hidden) in the service's online databases
  In the Virus Total case - the attacker needs a Virus Total subscription

VIRUSTOTAL DEMONSTRATION

As said before, the exfiltration requires two general implementation components:

- A way to hide the desired data in the satellite
- A way to "pick up" the exact satellite from the sandbox service database

These 2 parts create a wide range of options for implementation. We will demonstrate a couple of combinations, that vary in both their ease of implementation, and their stealth level.

In all cases, let's assume that the data we want to exfiltrate is the following string:

```
MESSAGE IN A BOTTLE
```

All code snippets and other helpful content can be found in the SafeBreach-Labs/blog-snippets github repository.

Virus Total

Method #1: Magic String using spacebin

Concatenating a magic string

We chose to use the following simple struct for packing our desired data:

```
_$_MAGICSTR_$_HAA=TUVTU0FHRSBJTiBBIEJPVFRMRQ==
```

Where the message to exfiltrate (base64 encoded) follows its encoded length (base64 encoded), that follows a plain magic string. In the example above, the encoded message length is 0x1C, and the plain message before encoding is "MESSAGE IN A BOTTLE".

Note that one could choose to not only encode the message, but also encrypt it using a symmetric key that will later be used to decrypt it.

Another more complicated packing suggestion is to prefix the message with N encrypted bytes, followed by those same N bytes, unencrypted. After acquiring the sample, we could scan it sequentially and decrypt any sequence of bytes, then check the decryption result against the next N bytes. If a match is found, this means that the following bytes are the desired exfiltrated data. This model would allow one to use asymmetric keys for encryption and decryption.

Using SafeBreach-Labs/spacebin

We use the simple go.bat to incorporate our pre-prepared string into a templated executable.

```
>go.bat /W:"_$_MAGICSTR_$_HAA=TUVTU0FHRSBJTiBBIEJPVFRMRQ=="
```

Creating a YARA Rule

We add a new YARA rule to match samples with our magic string:

```
rule magic_string
{
    meta:
        description = "Samples that contain our magic string"

    strings:
        $ms = "_$_MAGICSTR_$_"

    condition:
        $ms
}
```

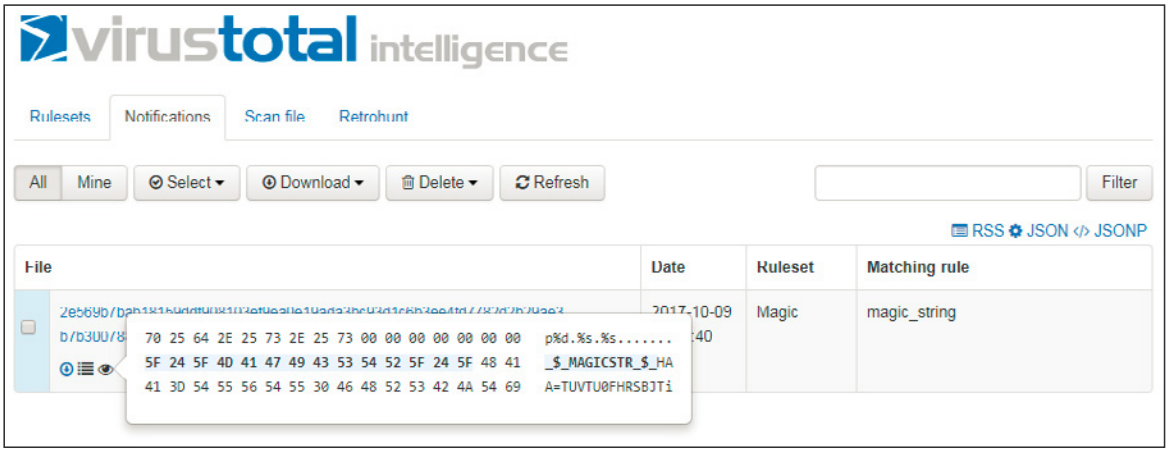
Uploading to VirusTotal

We then upload the executable to VirusTotal , to demonstrate uploading of the satellite that triggered an AV product in the target network. Our uploaded sample hash:

```
2e569b7bab18159ddf908103ef9ea0e19ada3bc93d1c6b3ee4fd7782d2b29ae3
```

Downloading Samples

We get live notifications from VirusTotal about new samples that match our YARA rule, and we download any new sample:



Extracting Data

All we got left to do is find the magic string, and extract the desired data from the struct, using the following function:

```
def extract_packed_string(magic_str, bin_data):
    magic_offset = bin_data.find(magic_str)
    length_offset = magic_offset + len(magic_str)
    data_length = struct.unpack("<H",
        b64decode(bin_data[length_offset:length_offset+4]))[0]
    return b64decode(bin_data[length_offset+4:length_offset+4+data_length])
```

And we get the original message:

```
MESSAGE IN A BOTTLE
```

Method #2: Data inside a Well-Known Malware

Now let us go one level up to another method, that shows a more covert approach. The following method shows how the attacker, as a VirusTotal subscriber, can perform less targeted queries to get his desired sample. This will distance the attacker from the satellite samples, making it harder to track his/her identity, even if we assume that VirusTotal itself monitors its subscriber behavior for forensics and investigation.

Finding an Appropriate Malware

The malware we want to piggyback on should be a part of a family that is: known enough to have many users searching and downloading it on the one hand; but specific enough to avoid exhaustion of the VirusTotal subscription limit.
We base this method on VirusTotal special search modifier - "similar-to". We have minimal knowledge of how VirusTotal classifies executables as "similar" to each other, so we want to make the smallest change possible to the original sample.

We chose CryptoWall[1] for this job. We took one of its identified samples, and looked for similar binaries, using this VirusTotal intelligence search:

```
similar-to:bf352825a70685039401abde5daf1712fd968d6eee233ea72393cbc6faffe5a2
```

The results showed 5 samples, with upload dates ranging from end of 2015 to end of 2017.

Inserting Data String into the Sample

Inserting the data string to the sample can be achieved in many ways. The sample is an executable so we chose to use the PE structure for that matter. Let us present several alternatives:

Adding String as Longest in PE

One example of how it can be done is to add a null terminated string to the ".rdata" section, just after all other existing strings. We use the length trick to avoid including the embedded, distinguishable magic string:
we pad the desired data with enough characters, to make it the longest string in the PE. It can be easily calculated using SysInternals strings.exe utility, as described below.

Iterate i from 0 until no strings are found:

```
>strings.exe -n i bf352825a70685039401abde5daf1712fd968d6eee233ea72393cbc6faffe5a2
```

We finished with i = 337.
In our case, we concluded that our string should be a UNICODE of at least 338 chars long.

[1] https://researchcenter.paloaltonetworks.com/2015/11/cryptowall-v4-emerges-days-after-cyber-threat-alliance-report/

In future development, if one wants to support bigger exfiltration capacity, we could break the desired data into multiple strings of the same length, maintaining each of them as longest in PE.

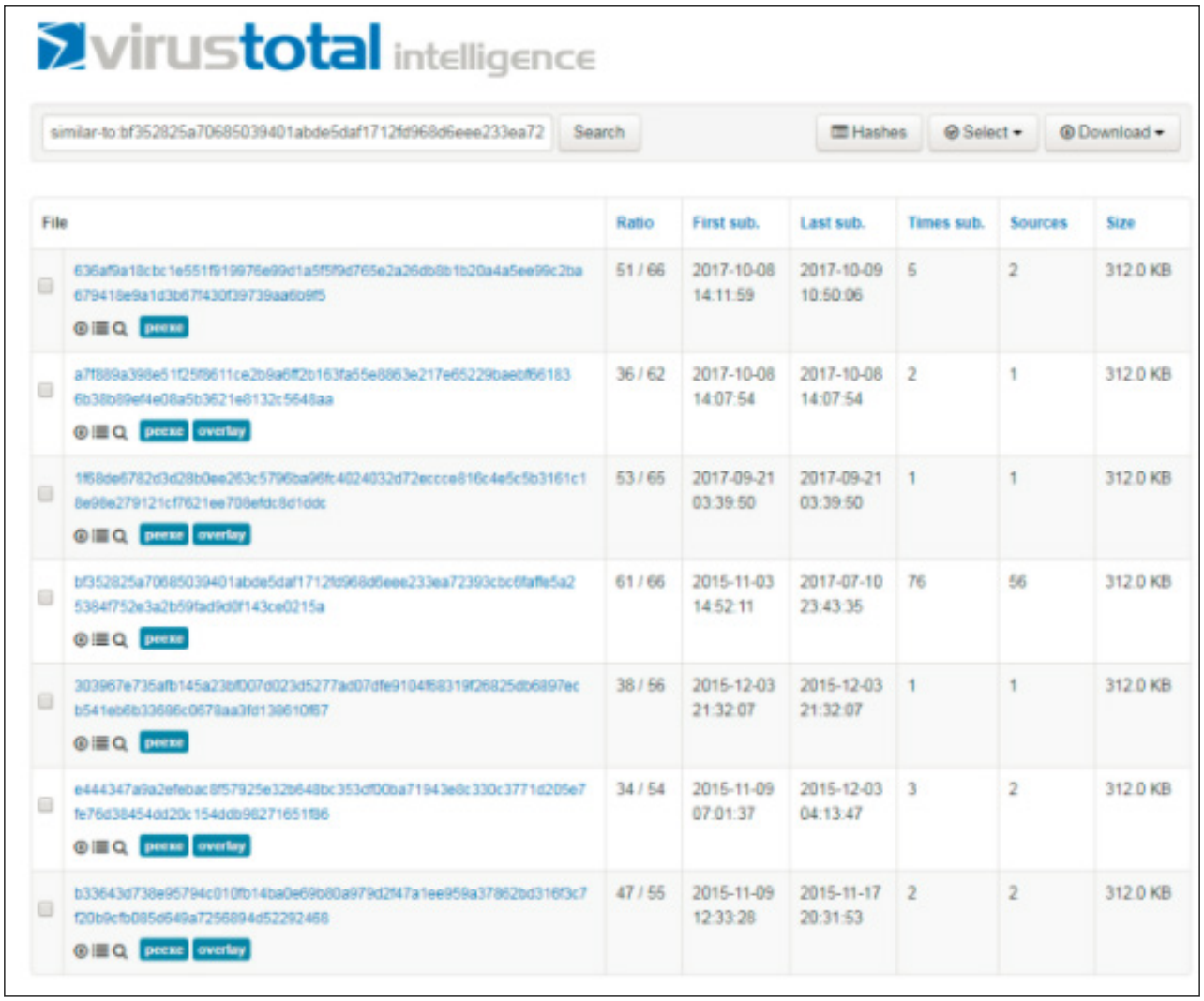
After the string is prepared, we push it into the trailing null area that's in the rdata section using the script here, that uses the pefile module to manipulate the PE.

Adding String as a PE Resource

Another way is by adding a string resource that contains our desired data. We will either want to make it the longest as described before, or, alternatively - place it under a specific resource ID that we will look for after downloading the sample. There are many tools available for this purpose.

Downloading Samples

Now (after the satellite was uploaded to VirusTotal), all we have to do is make the same "similar-to" query we performed earlier, and download all the samples. Because we didn't change much of the PE structure, we will see our own crafted sample in one of the recent results. We will download all the samples to keep on hiding our true target sample:



On our own computer, we will extract the longest string out of each sample, and try to decode each of them. One of them will succeed - and this one is of course our desired sample. Again, we got the desired data!

Adding String as a New Section

We suggest another way to include the string in the PE by adding a new section with a distinguishable name. It does not have the "similar-to" benefits that allow us to hide better in the sea of VirusTotal queries. This time we use the VirusTotal "section:" search modifier. First we make a search to make sure no other samples have the proposed name:

```
section:".EXFILSECTION"
```

Now we edit the original sample PE to add a whole new section named ".EXFILSECTION". If the desired sample has been uploaded from the target network, performing the same query will show our crafted sample. Open and decode all the data that's in the new section. That's the data we exfiltrated.

HYBRID ANALYSIS DEMONSTRATION

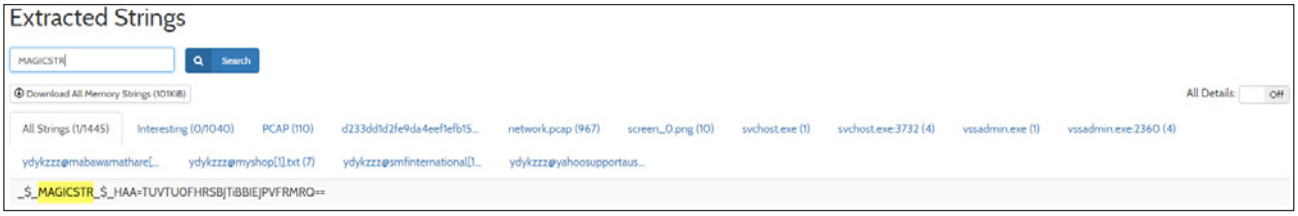
We will demonstrate how similar techniques can be performed on another popular sandbox service - www.hybrid-analysis.com. The playing ground in Hybrid Analysis differs from VirusTotal in 2 main aspects: Hybrid Analysis does not require subscription[2] for performing advanced searches and downloading samples; Hybrid Analysis has a different (a bit less comprehensive) search options. These 2 differences do not prevent from performing a similar exfiltration scheme such as #2 that is described above.

We could even perform the same "similar-to" search, using the same CryptoWall sample as the family representative:

- Initially, this search resulted in 19 samples ranging from October 16 to November 17.
- We then insert the message string to the PE using one of the manners explained above (in the VirusTotal case). For the purpose of this demonstration, we chose to simply insert the message data with a magic string into the PE's rdata section, using the same packing format as described above.
- Now, the same "similar-to" query returns 20 results.

[2] Hybrid-analysis has recently introduced a new vetting process - each user must be approved in order to enable features such as downloading malware samples. This requires filing a request and providing identification details, that optionally prove the user's integrity as a security professional. In some of the cases we demonstrated here there is no need to actually download the sample, so this new policy may only partially limit the attacker's range of options, if at all.

And in the sample details we can see our crafted exfiltration string. We may choose to download the sample and passively look for it, or just use the web interface to acquire it:



CONCLUSION

Public sandbox services that allow both upload and search capabilities may be used as a means for data exfiltration. The database for these services is an intermediary for transferring hidden data from a source machine to an attacker who is looking for the expected data. Many permutations of this exfiltration model may be created - each features a different stealth level, ease of implementation, accuracy, capacity etc. We only demonstrated a couple of them.