Repurposing OnionDuke: A Single Case Study Around Reusing Nation State Malware

Josh Pitts
the.midnite.runr@gmail.com
Black Hat USA 2015

Abstract- **Nation State malware and tools are not magical. However, they are effective because they are developed in private, have a budget, and maybe teams of engineers. Whenever one of these digital weapons is released to the public, discovered via forensics or an information leak, it allows all interested parties to learn and improve either their defensive or offensive capabilities. This paper inspects the OnionDuke packer discovered by the author in October 2014; with the result of repurposing it in the author's own tool set.**

I. INTRODUCTION

Repurposing is what we do as humans. As Picasso said, "Good artists copy, great artists steal." This is why we have trademarks, patents, and copyrights. When it comes to digital weapons developed by nation states and criminals, there is little risk in terms of legal retribution (outside of criminal prosecution and other forums of retaliation) for researching and reusing ideas or components.

Repurposing by nation states and criminals

The Sony attack reported in 2014 [1] featured a destructive malware named *Destover*. *Destover's* main feature was delayed hard drive erasing. North Korea was ultimately blamed for this attack because Sony's comedy movie *The Interview* apparently told an insulting story of the Democratic People's Republic of Korea's leader. The *Destover* malware shared similar command and control servers as *Volgmer* against South Korea in 2014, also linked to North Korea. *Destover* had similar file names as *Jokra* also used against South Korea in 2013 and again linked to North Korea. In addition, *Destover* had the same non-malicious drivers as those used in the *Shamoon* attacks, reported on August 16, 2012, against energy and oil companies by the group named *Cutting Sword of Justice* [2].

According to documents dated June 2012, the NSA developed their own hard drive data destruction malware labeled *PITIEDFOOL* [3]. However, the idea of destructive malware is not new, the most famous probably being the *Michelangelo virus* [4], first discovered in early 1991.

The host based security firm, Kaspersky, documented the activities of the EQUATION Group in February 2015 [5]. While the report does not mention the NSA directly, there are instances where they are similar. In 2009, Google asked the NSA for help with the Aurora intrusion [6]. In the report released by Kaspersky, the 2009 Aurora exploit was used in Afghanistan by the EQUATION Group (CVE-2013-3918). Meaning the EQUATION Group recovered and repurposed the exploit from the Aurora intrusion. In addition, two Stuxnet exploits were used by the EQUATION Group, MS09-025 and CVE-2010-2568.

Through leaked documents by Edward Snowden, *Der Spiegel* [7][8] reported that in 2012 the NSA used and repurposed South Korean implants on North Korean networks. The NSA leveraged existing botnets and command and control networks to deploy implants and collect information. Additionally, the NSA captured zero day exploits in passive collection, possibly for reuse.

Criminals have a different motive than nation states, as their primary purpose of activity is the creation of revenue [9]. Criminals will usually take the quickest route to achieve that goal while reusing openly available tools and exploits. From a CyActive report titled *Cyber Security's "Infamous Five" of 2014* [10], five types of malware are discussed with specifics around the number of reused components. Of the five, the least amount of reused components used was four; with the greatest amount of reused components used were 12.

On July 6th 2015, HackingTeam Remote Control System and associated source code, zero day exploits, emails, and internal documents were dumped via bittorrent and made available on the Internet [11]. Within days a HackingTeam zero day flash exploit was included in the Angler Exploit Kit [12].

II. OnionDuke Discovery

Searching for the Kraken

During the author's 2014 DerbyCon Talk, *A Year in the Backdoor Factory* [13], it was stated that he believed that binaries were being patched during download on the Internet. Tor Proxy and Exitmap were used to find a malicious Tor exit node MITM patching binaries during download and the details were outlined in the blog post *The Case of the Modified Binaries* [14]. F-Secure named the deployed malware *OnionDuke* because of its relation to CosmicDuke and MiniDuke and its suspected links to the Russian Government [15].

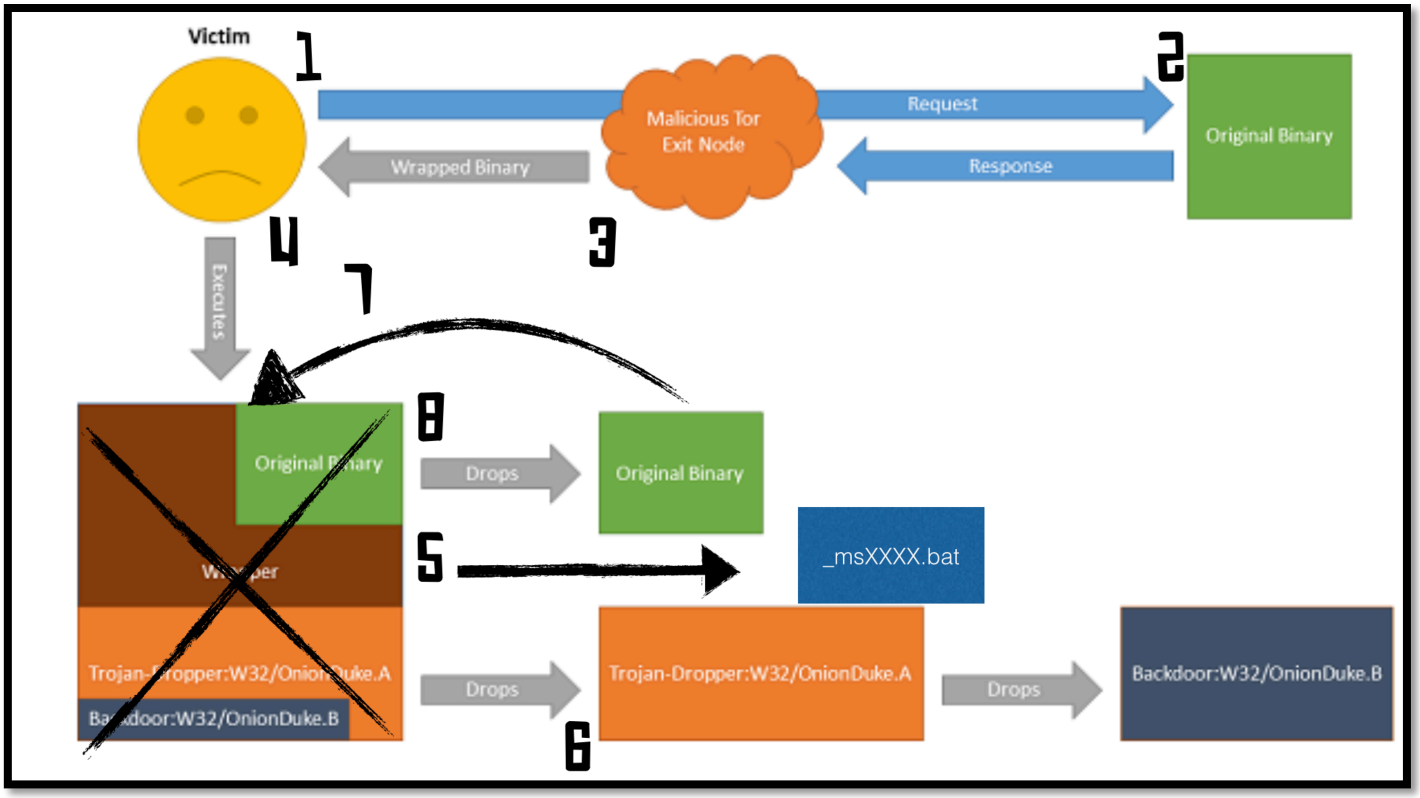
III. OnionDuke Packer

Understanding the Packer

From the author's background in developing *The Backdoor Factory* (BDF) and *BDFProxy* [16], the most interesting part of OnionDuke is the packer, not the dropped malware itself.

The OnionDuke packer was surprisingly simple. Fig 1, originally from F-secure, outlines the steps that are taken from initial download to infection. Step one and two; the user downloads a PE binary over HTTP through a Malicious Tor exit node, or any node on the Internet. As the binary moves through the malicious exit node (step three), it is wrapped/patched with the unpacking stub and malware. Next the user executes (step four) the binary. Upon execution, the OnionDuke unpacking stub drops a batch file, original binary (originalname.exe.org), and malware (file.exe) in the user's %temp% directory and executes the batch file (step 5). Next the malware is executed (step 6). After the malware has infected the machine, the batch file overwrites the packed file with the original binary (step 7). Then the batch file executes the original binary (step 8) and deletes the original file and itself in the %temp% directory. This approach is advantageous if the user believes that something is incorrect with the downloaded file and chooses to look for an expected signature or hash of the file after the initial execution – as the downloaded file will be restored to the expected original binary.

Figure 1: OnionDuke Unpack Order



Static Analysis

The author captured nine samples over over five days. During that time, he periodically re-downloaded the same binaries through the malicious exit node to determine if there were changes in the packing/wrapping method. There were no noted differences from binary to binary. All patched/wrapped binaries deployed malware as a PE executable and not a DLL.

Upon manual inspection of the section hashes from VirusTotal [17][18][19], it is noted that the text, rdata, and reloc sections each share the same section hashes between samples collected. The data sections are the same size but have different hashes. By cutting out the data sections between different OnionDuke files and comparing them, the differences were found between offsets 0xfb20 and 0xfd6c. The first main difference between data sections was the storage of original file name formatted as originalfilename.exe.org, located at offset 0xfb20. There is extraneous data included at the end of an OnionDuke binary, as the physical size is greater than the virtual size that is loaded into memory. This appended data is associated with offset 0xfc28, which is a four-byte pointer to the beginning of the original file located on disk. The next four bytes is the size of the original file. It should be noted that by comparing the appended size to the original file size that the appended file is compressed. At file offset 0xfd3c is the four-byte pointer to the location on disk of the appended OnionDuke Malware (file.exe named at offset 0xfc34) also in a compressed format with the size denoted by the next four bytes.

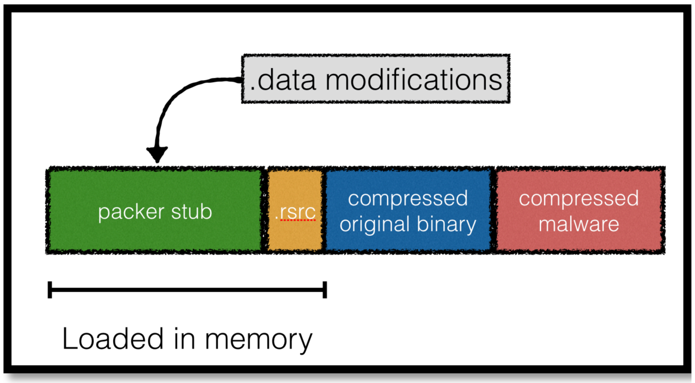
Stealing Resources

Felix Grobet, et al, presented a MITM patching system named *cyanid and calcium* in 2008 [20] where a noted drawback was that the associated file icons would not be what the user would be potentially expecting and thus warning the user that something was suspicious. However, OnionDuke solves this issue by taking the original binary's resource section (rsrc), removing excess padding, and updating the RVA of the pointers to each resource so that the PE loads correctly and looks the same on disk. The author's tool BDF, never had this issue as code/payloads are patched into the host binary and resources stay intact.

Stub Details

The OnionDuke packer format is noted in Fig 2. The packer stub includes text, data, rdata, and reloc sections.

Figure 2: OnionDuke Packer Layout



Modifications are made to the data section based on size and location of the original binary and malware. The rsrc section is taken from the original binary and modified. The original binary is compressed and XOR'ed. The stub appears to be written in C++, compiled with Visual Studio with the Buffer Security Check (security cookie), supports both ANSI and Unicode file and path names, and captures command line arguments to pass back to the original program at execution.

XOR and Compression

Before decompressing the appended compressed binaries to disk, the data blobs are each XOR'ed with the same four-byte key. The XOR key is located in two locations in the text section at offsets 0x413 and 0x429 respectively.

The compression algorithm applied can be recognized by its magic number AP32, which is for the aPLib compression library written by Jørgen Ibsen [21]. aPLib libraries can easily be imported into C/C++ for only x86/x64 projects.

Malware Deployment

All of the nine samples downloaded by the author deployed PE binaries. However, there existed two ways to deploy a DLL: Via rundll32 either by calling the DLL export of *printMessage* or by ordinal number. F-secure found one OnionDuke associated DLL, however, the packer was not recovered [22]. The author believes that the called export of *printMessage* or the ordinal values could be customized as needed based on the campaign by the authors. To denote if the deployed malware is a DLL, the byte at offset 0xfd38 is set to 0x01 and if an ordinal is used, the ordinal value will be set beginning at offset 0xfd48.

III. Repurposing the OnionDuke Packer

What to Keep?

While the detection rate for the associated packer is high at 41/55 AVs for one particular sample [23], the author believes all of the features and capabilities should be reused for an exercise to show how easy it is to repurpose nation state code/malware. Therefore the author will save the OnionDuke stub itself and make modifications to the stub to support BDF user provided binaries and malware.

Implementing in BDF/BDFProxy

BDF/BDFProxy includes a static PE parser and the framework in place to patch PE files. Therefore minimal changes were needed to implement this into the existing framework. The most difficult part of the process was building a resource section parser to fix up the original binary's rsrc section RVA values (Example: icon pointers) according to the OnionDuke Stub RVA offsets. Overall ~250 lines of code were added to BDF to support the OnionDuke patching/wrapping method.

As the anti-virus (AV) detection is high for the original packer, variability was added in a couple ways: Randomize the name of dropped binaries, randomize the XOR key used for the appended binaries, and add a random four byte number to each section to randomize the section hash.

Usage

Within 'the-backdoor-factory' directory execute the following command:

```
./backdoor.py –f filetopatch.exe –m onionduke –b malware.exe
```

BDF supports both win x86/x64 PE executable binaries and DLLs for the OnionDuke patching/wrapping method. The only caveat for DLLs, the DLL should export *printMessage*, use ordinal 0x01, or like a meterpreter DLL - execute no matter what ordinal or export is called.

IV. Mitigations

The best AV evasion that was achieved, without modifying large swaths of the text section, was 21/55 on VirusTotal [24]. This does not include the deployed malware that is written to disk during execution. However, many popular AVs missed this including Sophos, Microsoft, McAfee, Malwarebytes, Symantec, and Trendmicro. Greater AV evasion can be achieved by re-writing the packer in C/C++, however, that was not the purpose of this paper.

To prevent MITM patching of binaries software vendors and distribution sites should implement TLS with HTTP Strict Transport Security (HSTS). For host-based security, large enterprises should implement whitelisting and restrict the downloading of binaries. For individuals, check signatures if the binary is signed and verify that hashes match on download.

ACKNOWLEDGMENT

Special thanks to Travis Morrow, Matt Graeber, Jason Butterfield, Chris Truncer, Will Schroeder, and the crew at Leviathan Security.

REFERENCES

[1] Symantec, *Destover Destructive malware has links to attacks on South Korea*. http://www.symantec.com/connect/blogs/destover-destructive-malware-has-links-attacks-south-korea

[2] Wikipedia, *Shamoon Malware.* https://en.wikipedia.org/wiki/Shamoon

[3] PC World, *Report: NSA not only creates, but also hijacks, malware.* http://www.pcworld.idg.com.au/article/564189/report-nsa-only-creates-also-hijacks-malware/

[4] Wikipedia, *Michelangelo Virus*, https://en.wikipedia.org/wiki/Michelangelo_(computer_virus)

[5] Kaspersky, *EQUATION GROUP: QUESTIONS AND ANSWERS.* https://securelist.com/files/2015/02/Equation_group_questions_and_answers.pdf

[6] Wired, *Google Asks NSA to Help Secure its network.* http://www.wired.com/2010/02/google-seeks-nsa-help/

[7] Ars Technica, *NSA Secretly hijacked existing malware to spy on N. Korea, others* http://arstechnica.com/information-technology/2015/01/nsa-secretly-hijacked-existing-malware-to-spy-on-n-korea-others/

[8] Der Spiegel, http://www.spiegel.de/international/world/new-snowden-docs-indicate-scope-of-nsa-preparations-for-cyber-battle-a-1013409.html

[9] Kaspersky, *What Motivates Cybercriminals? Money, Of Course.* https://blog.kaspersky.com/what-motivates-cybercriminals-money-of-course/

[10] CyActive, *Cyber Security's Infamous Five of 2014.* http://www.cyactive.com/infamous-five-of-2014/

[11] Hacking Team documents, https://ht.transparencytoolkit.org/

[12] Malware Don't Need Coffee, *CVE-2015-5119 (HackingTeam 0d - Flash up to 18.0.0.194) and Exploit Kits.* http://malware.dontneedcoffee.com/2015/07/hackingteam-flash-0d-cve-2015-xxxx-and.html

[13] Joshua Pitts, *A Year in the Backdoor Factory. https://www.youtube.com/watch?v=LjUN9MACaTs*

[14] Joshua Pitts, *The Case of the Modified Binaries*. http://www.leviathansecurity.com/blog/the-case-of-the-modified-binaries

[15] The Guardian, *Evidence implicates government-backed hackers in Tor malware attacks.* http://www.theguardian.com/technology/2014/nov/14/government-hackers-tor-malware-attacks-onionduke-miniduke

[16] Joshua Pitts, *The Backdoor Factory and BDFProxy*. https://github.com/secretsquirrel/the-backdoor-factory

[17] VirusTotal, *An OnionDuke sample*. https://www.virustotal.com/en/file/de1a78b4a65d76d26f04db0c1fd5eefdb9361f434925df88e45d6cd511f3c013/analysis/

[18] VirusTotal, *An OnionDuke sample.* https://www.virustotal.com/en/file/4910e4a5e2eed444810c62a0e9a32affb8a41693b2fcff49aabd9c125fa796d1/analysis/

[19] VirusTotal, *An OnionDuke sample.* https://www.virustotal.com/en/file/a3e5b92ce574397000825dc646e1a7763b7f817bb8ac8d446a31c3252c1076eb/analysis/

[20] Felix Grobet, et al. *Software Distribution Malware Infection Vector.* https://dl.packetstormsecurity.net/papers/general/Software.Distribution.Malware.Infection.Vector.pdf

[21] Jørgen Ibsen, *aPLib compression library.* http://ibsensoftware.com/products_aPLib.html

[22] VirusTotal via F-Secure, *OnionDuke DLL Dropper*. https://www.virustotal.com/en/file/0102777ec0357655c4313419be3a15c4ca17c4f9cb4a440bfb16195239905ade/analysis/

[23] VirusTotal, *OnionDuke wrapper with procexp.exe.* https://www.virustotal.com/en/file/4910e4a5e2eed444810c62a0e9a32affb8a41693b2fcff49aabd9c125fa796d1/analysis/

[24] VirusTotal, *BDF using OnionDuke method*. https://www.virustotal.com/en/file/4ca772598eb65b915c65a928fe84fd33fa22fe37965729692591412 93a7ec35f/analysis/1437012262/