Mahalo FIN7: Responding to the Criminal Operators' New Tools and Techniques

fireeye.com/blog/threat-research/2019/10/mahalo-fin7-responding-to-new-tools-and-techniques.html

Breadcrumb

Threat Research

Nick Carr, Josh Yoder, Kimberly Goody, Scott Runnels, Jeremy Kennelly, Jordan Nuce

Oct 10, 2019

11 mins read

Threat Research

Malware

During several recent incident response engagements, FireEye Mandiant investigators uncovered new tools in FIN7's malware arsenal and kept pace as the global criminal operators attempted new evasion techniques. In this blog, we reveal two of FIN7's new tools that we have called BOOSTWRITE and RDFSNIFFER.

The first of FIN7's new tools is BOOSTWRITE – an in-memory-only dropper that decrypts embedded payloads using an encryption key retrieved from a remote server at runtime. FIN7 has been observed making small changes to this malware family using multiple methods to avoid traditional antivirus detection, including a BOOSTWRITE sample where the dropper was signed by a valid Certificate Authority. One of the analyzed BOOSTWRITE variants contained two payloads: CARBANAK and RDFSNIFFER. While CARBANAK has been thoroughly analyzed and has been used maliciously by several financial attackers including FIN7, RDFSNIFFER is a newly-identified tool recovered by Mandiant investigators.

RDFSNIFFER, a payload of BOOSTWRITE, appears to have been developed to tamper with NCR Corporation's “Aloha Command Center” client. NCR Aloha Command Center is a remote administration toolset designed to manage and troubleshoot systems within payment card processing sectors running the Command Center Agent. The malware loads into the same process as the Command Center process by abusing the DLL load order of the legitimate Aloha utility. Mandiant provided this information to NCR.

BOOSTWRITE Loader: Where You At?

BOOSTWRITE is a loader crafted to be launched via abuse of the DLL search order of applications which load the legitimate ‘Dwrite.dll’ provided by the Microsoft DirectX Typography Services. The application loads the ‘gdi’ library, which loads the ‘gdiplus’ library, which ultimately loads ‘Dwrite’. Mandiant identified instances where BOOSTWRITE was placed on the file system alongside the RDFClient binary to force the application to import DWriteCreateFactory from it rather than the legitimate DWrite.dll.

Once loaded, `DWrite.dll` connects to a hard-coded IP and port from which it retrieves a decryption key and initialization vector (IV) to decrypt two embedded payload DLLs. To accomplish this task, the malware first generates a random file name to be used as a text log under the current user's %TEMP% directory; this filename starts with ~rdf and is followed by a set of random numbers. Next, the malware scans its own image to find the location of a 32-byte long multi-XOR key which is used to decode data inside its body. Part of the decoded data is an IP address and port which are used to retrieve the key and the IV for the decryption of the embedded payloads. The encryption algorithm uses the ChaCha stream cipher with a 256-bit key and 64-bit IV.

Once the key and the IV are downloaded the malware decrypts the embedded payloads and performs sanity checks on the results. The payloads are expected to be PE32.DLLs which, if the tests pass, are loaded into memory without touching the filesystem.

The malware logs various plaintext messages to the previously created logfile %TEMP%\~rds<rnd_numbers> which are indicative of the loader’s execution progress. An example of the file content is shown in Figure 1:

```
Loading...
Starting...
Init OK
Key OK
Data: 4606941
HS: 20
K:[32] V:[8]
DCnt: 732642317(ERR)
```

Figure 1: BOOSTWRITE log file

Before exiting, the malware resolves the location of the benign DWrite.dll library and passes the execution control to its DWriteCreateFactory method.

The malware decrypts and loads two payload DLLs. One of the DLLs is an instance of the CARBANAK backdoor; the other DLL is a tool tracked by FireEye as RDFSNIFFER which allows an attacker to hijack instances of the NCR Aloha Command Center Client application and interact with victim systems via existing legitimate 2FA sessions.

RDFSNIFFER Module: We Smell a RAT

RDFSNIFFER is a module loaded by BOOSTWRITE which allows an attacker to monitor and tamper with legitimate connections made via NCR Corporation’s ‘Aloha Command Center Client’ (RDFClient), an application designed to provide visibility and system management capabilities to remote IT techs. RDFSNIFFER loads into the same process as the legitimate RDFClient by abusing the utility’s DLL load order, launching each time the ‘Aloha Command Center Client’ is executed on an impacted system.

When the RDFSNIFFER module is loaded by BOOSTWRITE it hooks several Win32 API functions intended to enable it to tamper with NCR Aloha Command Center Client sessions or hijack elements of its user-interface (Table 1). Furthermore, this enables the malware to alter the user’s last input time to ensure application sessions do not time out.

Win32 API Function	Hook Description
CertVerifyCertificateChainPolicy	Used to man-in-the-middle SSL sessions
CertGetCertificateChain	Used to man-in-the-middle SSL sessions
WSAConnect	Used to man-in-the-middle socket connections
connect	Used to man-in-the-middle socket connections
ConnectEx	Used to man-in-the-middle socket connections
DispatchMessageW	Used to hijack the utility's UI
DispatchMessageA	Used to hijack the utility's UI
DefWindowProcW	Used to hijack the utility's UI
DefWindowProcA	Used to hijack the utility's UI
GetLastInputInfo	Used to change the user's last input time (to avoid timed lock outs)

Table 1: RDFSNIFFER's Hooked Win32 API Functions

This module also contains a backdoor component that enables it to inject commands into an active RDFClient session. This backdoor allows an attacker to upload, download, execute and/or delete arbitrary files (Table 2).

Command Name	Legit Function in RDFClient	RDFClient Command ID	Description
Upload	FileMgrSendFile	107	Uploads a file to the remote system
Download	FileMgrGetFile	108	Retrieves a file from the remote system
Execute	RunCommand	3001	Executes a command on the remote system
DeleteRemote	FileMgrDeleteFile	3019	Deletes file on remote system
DeleteLocal	-	-	Deletes a local file

Table 2: RDFSNIFFER's Backdoor Functions

Signed: Yours Truly, FIN7

While the majority of BOOSTWRITE variants recovered from investigations have been unsigned, Mandiant identified a signed BOOSTWRITE sample used by FIN7 during a recent investigation. Following that discovery, a signed BOOSTWRITE sample was uploaded to VirusTotal on October 3. This executable uses a code signing certificate issued by MANGO ENTERPRISE LIMITED (Table 3).

MD5	Organization	Country	Serial
a67d6e87283c34459b4660f19747a306	mango ENTERPRISE LIMITED	GB	32 7F 8F 10 74 78 42 4A BE B8 2A 85 DC 36 57 03 CC 82 70 5B

Table 3: Code signing certificate used for BOOSTWRITE

This indicates the operators may be actively altering this malware to avoid traditional detection mechanisms. Notably, the signed BOOSTWRITE sample had a 0/68 detection ratio when it was uploaded to VirusTotal, demonstrating the effectiveness of this tactic (Figure 2).

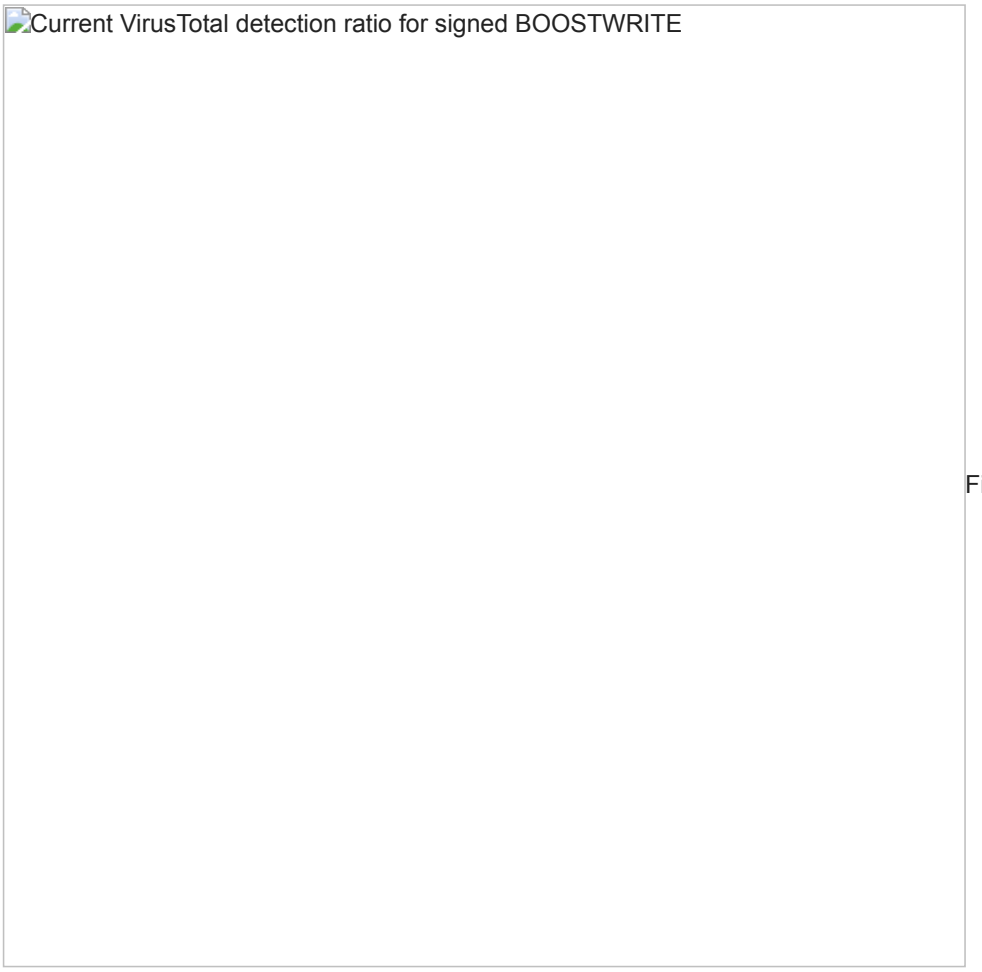

Figure 2: Current VirusTotal detection ratio for signed BOOSTWRITE

Use of a code signing certificate for BOOSTWRITE is not a completely new technique for FIN7 as the group has used digital certificates in the past to sign their phishing documents, backdoors, and later stage tools. By exploiting the trust inherently provided by code certificates, FIN7 increases their chances of bypassing various security controls and successfully compromising victims. The full evasion achieved against the detection engines deployed to VirusTotal – as compared to an unsigned BOOSTWRITE sample with an invalid checksum– illustrates that FIN7's methods were effective in subverting both traditional detection and ML binary classification engines. This is a known issue and has been deeply studied since at least 2016's "Chains of Distrust" research and 2017's "Certified Malware" paper. Since there are plenty of goodware samples with bad or no signatures – and a growing number of malware samples with good signatures – there is no easy solution here. The upside is that vendors selectively deploy engines to VirusTotal (including FireEye) and VT detection performance often isn't a comprehensive representation of encountering full security technology stacks that implement detection-in-depth. Later in this blog we further explore BOOSTWRITE's PE Authenticode signature, its anomalies, and how code signing can be turned from a detection challenge into detection opportunities.

Outlook and Implications

While these incidents have also included FIN7's typical and long-used toolsets, such as CARBANAK and BABYMETAL, the introduction of new tools and techniques provides further evidence FIN7 is continuing to evolve in response to security enhancements. Further, the use of code signing in at least one case highlights the group's judicious use of resources, potentially limiting their use of these certificates to cases where they have been attempting to bypass particular security controls. Barring any further law enforcement actions, we expect at least a portion of the actors who comprise the FIN7 criminal organization to continue conducting campaigns. As a result, organizations need to remain vigilant and continue to monitor for changes in methods employed by the FIN7 actors.

Sigs Up Dudes! Indicators, Toolmarks, and Detection Opportunities

While FireEye does not release our production detection logic for the code families, this section does contain some identification and hunting concepts that we adopt in our layered detection strategy. Table 4 contains malware samples referenced in this blog that FireEye is able to share from the larger set recovered during active investigations.

Type	Indicator(s)
BOOSTWRITE (signed)	MD5: a67d6e87283c34459b4660f19747a306 SHA-1: a873f3417d54220e978d0ca9ceb63cf13ec71f84 SHA-256: 18cc54e2fbdad5a317b6aeb2e7db3973cc5ffb01bbf810869d79e9cb3bf02bd5 C2: 109.230.199[.]227
BOOSTWRITE (unsigned)	MD5: af2f4142463f42548b8650a3adf5ceb2 SHA1: 09f3c9ae382fbd29fb47ecdfeb3bb149d7e961a1 SHA256: 8773aeb53d9034dc8de339651e61d8d6ae0a895c4c89b670d501db8dc60cd2d0 C2: 109.230.199[.]227

Table 4: Publicly-shareable BOOSTWRITE samples

The signed BOOSTWRITE sample has a PE Authenticode anomaly that can be detected using yara's PE signature module. Specifically, the PE linker timestamp is prior to the Authenticode validity period, as seen in Table 5.

Timestamp	Description
2019-05-20 09:50:55 UTC	Signed BOOSTWRITE's PE compilation time
2019-05-22 00:00 UTC through 2020-05-21 23:59 UTC	Signed BOOSTWRITE's "mango ENTERPRISE LIMITED" certificate validity window

Table 5: Relevant executabe timestamps

A public example of a Yara rule covering this particular PE Authenticode timestamp anomaly is available in a blog post from David Cannings, with the key logic shown in Figure 3.

```
pe.number_of_signatures > 0 and not for all i in (0..pe.number_of_signatures - 1):
    pe.signatures[i].valid_on(pe.timestamp)
```

Figure 3: Excerpt of NCC Group's research Yara rule

There are other PE Authenticode anomalies that can also be represented as Yara rules to surface similarly suspicious files. Of note, this signed BOOSTWRITE sample has no counter signature and, while the unauthenticated attributes timestamp structure is present, it is empty. In preparing this blog, FireEye's Advanced Practices team identified a possible issue with VirusTotal's parsing of signed executable timestamps as seen in Figure 4.

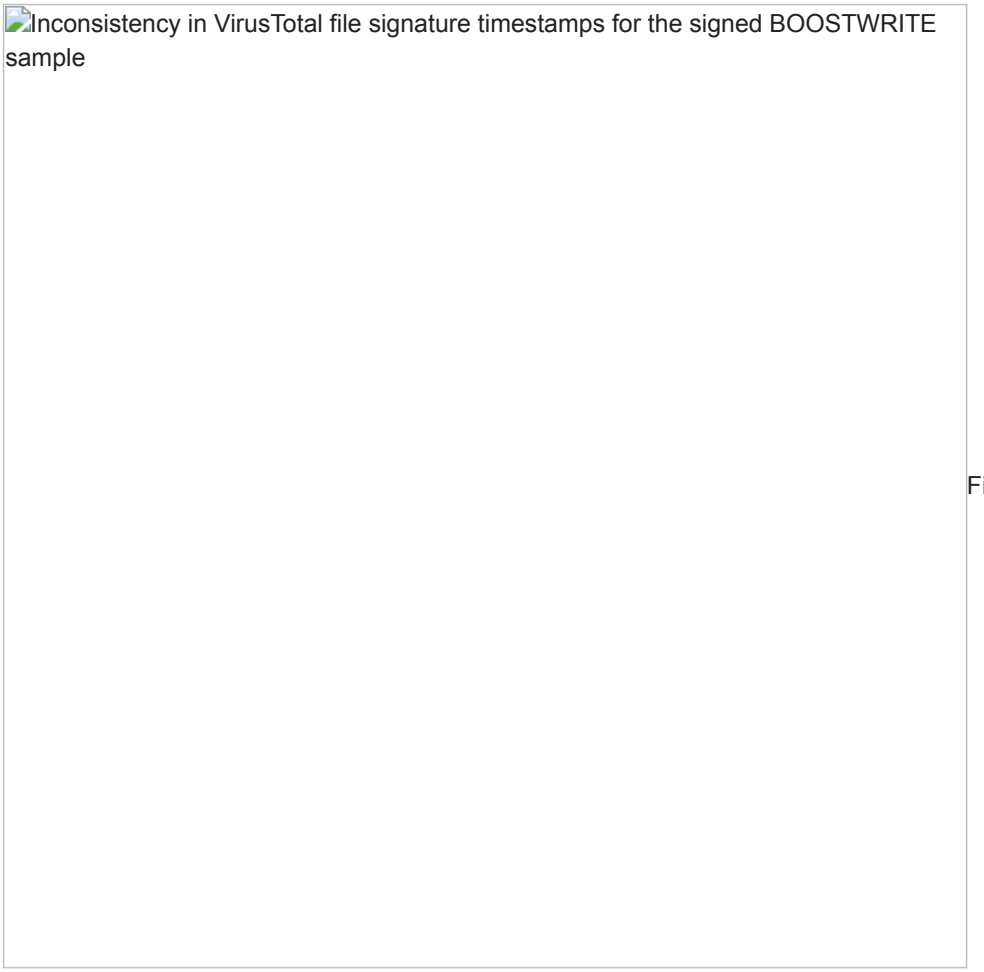

Figure 4: Inconsistency in VirusTotal file signature timestamps for the signed BOOSTWRITE sample

FireEye filed a bug report with Google to address the discrepancy in VirusTotal in order to remove confusion for other users.

To account for the detection weaknesses introduced by techniques like code signing, our Advanced Practices team combines the malicious confidence spectrum that comes from ML detection systems with file oddities and anomalies (weak signals) to surface highly interesting and evasive malware. This technique was recently described in our own Dr. Steven Miller's Definitive Dossier of Devilish Debug Details. In fact, the exact same program database (PDB) path-based approach from his blog can be applied to the toolmarks seen in this sample for a quick hunting rule. Figure 5 provides the PDB path of the BOOSTWRITE samples from this blog.

```
F:\projects\DWriteImpl\Release\DWriteImpl.pdb
```

Figure 5: BOOSTWRITE PDB path

The Yara rule template can be applied to result in the quick rule in Figure 6.

```
rule ConventionEngine_BOOSTWRITE
{
 meta:
    author = "Nick Carr (@itsreallynick)"
    reference = "https://www.fireeye.com/blog/threat-research/2019/08/definitive-dossier-of-devilish-debug-details-part-one-pdb-paths-malware.html"
 strings:
    $weetPDB = /RSDS[\x00-\xFF]{20}[a-zA-Z]?:?\\[\\\s|*\s]?.{0,250}\\DWriteImpl[\\\s|*\s]?.{0,250}\.pdb\x00/ nocase
 condition:
    (uint16(0) == 0x5A4D) and uint32(uint32(0x3C)) == 0x00004550 and $weetPDB and filesize < 6MB
}
```

Figure 6: Applying BOOSTWRITE's PDB path to a Yara rule

We can apply this same concept across other executable traits, such as BOOSTWRITE's export DLL name (DWriteImpl.dll), to create quick and easy rules that can aid in quick discovery as seen in Figure 7.

```
rule Exports_BOOSTWRITE
{
meta:
    author = "Steve Miller (@stvemillertime) & Nick Carr (@itsreallynick)"
strings:
    $exyPants = "DWriteImpl.dll" nocase
condition:
    uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and $exyPants at pe.rva_to_offset(uint32(pe.rva_to_offset(pe.data_directories[pe.IMAGE_DIRECTORY_ENTRY_EXPORT].virtual_address) + 12)) and filesize < 6MB
}
```

Figure 7: Applying BOOSTWRITE's export DLL names to a Yara rule (Note: this rule was updated following publication. It previously read "module_ls.dll", which is for Turla and unrelated.)

Of course, resilient prevention capabilities are needed and to that end, FireEye detects this activity across our platforms. Table 6 contains several specific detection names from a larger list of detection capabilities that captured this activity natively.

Platform	Signature Name
Endpoint Security	MalwareGuard ML detection (unsigned variants)
Network Security and Email Security	Malware.binary.dll (dynamic detection) MalwareGuard ML detection (unsigned variants) APTFIN.Dropper.Win.BOOSTWRITE (network traffic) APTFIN.Backdoor.Win.RDFSNIFFER (network traffic) FE_APTFIN_Dropper_Win_BOOSTWRITE (static code family detection) FE_APTFIN_Backdoor_Win_RDFSNIFFER (static code family detection)

Table 6: FireEye detection matrix

Don't Sweat the Techniques – MITRE ATT&CK Mappings

BOOSTWRITE

ID	Tactic	BOOSTWRITE Context

T1022	Data Encrypted	BOOSTWRITE encodes its payloads using a ChaCha stream cipher with a 256-bit key and 64-bit IV to evade detection
T1027	Obfuscated Files or Information	BOOSTWRITE encodes its payloads using a ChaCha stream cipher with a 256-bit key and 64-bit IV to evade detection
T1038	DLL Search Order Hijacking	BOOSTWRITE exploits the applications' loading of the 'gdi' library, which loads the 'gdiplus' library, which ultimately loads the local 'Dwrite' dll
T1116	Code Signing	BOOSTWRITE variants were observed signed by a valid CA
T1129	Execution through Module Load	BOOSTWRITE exploits the applications' loading of the 'gdi' library, which loads the 'gdiplus' library, which ultimately loads the local 'Dwrite' dll
T1140	Deobfuscate/Decode Files or Information	BOOSTWRITE decodes its payloads at runtime using using a ChaCha stream cipher with a 256-bit key and 64-bit IV

RDFSNIFFER

ID	Tactic	RDFSNIFFER Context
T1106	Execution through API	RDFSNIFFER hooks several Win32 API functions intended to enable it to tamper with NCR Aloha Command Center Client sessions or hijack elements of its user-interface
T1107	File Deletion	RDFSNIFFER has the capability of deleting local files
T1179	Hooking	RDFSNIFFER hooks several Win32 API functions intended to enable it to tamper with NCR Aloha Command Center Client sessions or hijack elements of its user-interface

Acknowledgements

The authors want to thank Steve Elovitz, Jeremy Koppen, and the many Mandiant incident responders that go toe-to-toe with FIN7 regularly, quietly evicting them from victim environments. We appreciate the thorough detection engineering from Ayako Matsuda and the reverse engineering from FLARE's Dimiter Andonov, Christopher Gardner and Tyler Dean. A special thanks to FLARE's Troy Ross for the development of his PE Signature analysis service and for answering our follow-up questions. Shout out to Steve Miller for his hot fire research and Yara anomaly work. And lastly, the rest of the Advanced Practices team for *both* the unparalleled front-line FIN7 technical intelligence expertise and MITRE ATT&CK automated mapping project – with a particular thanks to Regina Elwell and Barry Vengerik.