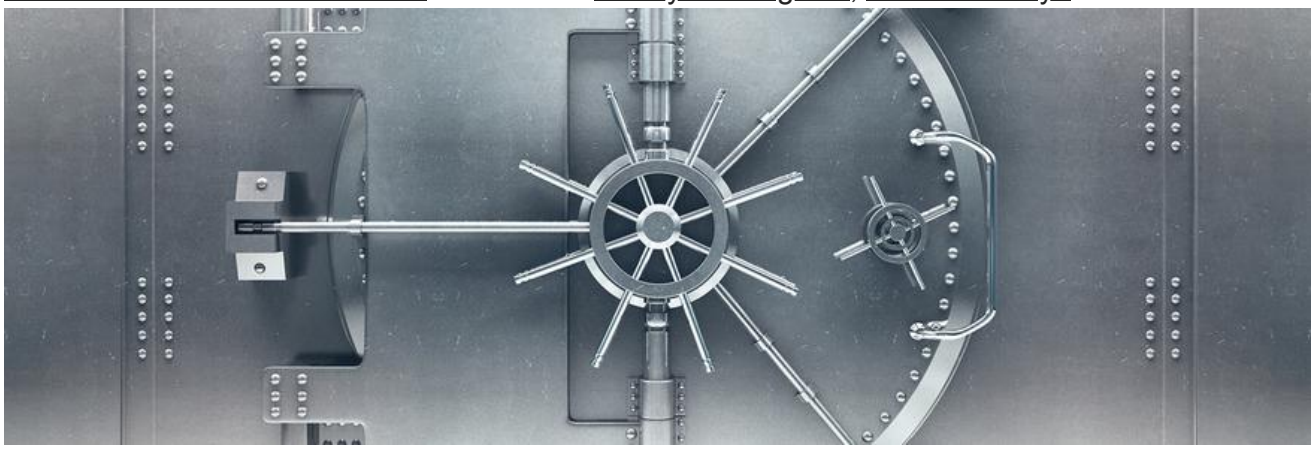
Threat Spotlight: Gootkit Banking Trojan

blogs.blackberry.com/en/2020/04/threat-spotlight-gootkit-banking-trojan

Masaki Kasuya, Tatsuya Hasegawa

RESEARCH & INTELLIGENCE / 04.13.20 / Tatsuya Hasegawa, Masaki Kasuya



Gootkit is a sophisticated banking Trojan which can perform various malicious activities such as: web injection, taking screenshots, video recording, email parsing, and so on. Gootkit emerged during the summer of 2014 but is still active, making it a viable threat to financial institutions to this day.

BlackBerry most recently observed a Gootkit campaign via AZORult infostealer malware in February, March, and April of 2019. Our monitoring revealed the threat actor changed Gootkit hosting domain names constantly and created Gootkit variants almost daily. Its core module contains several JavaScript files and the node.js runtime environment, so as a result, its file size tends to be large. In fact, our analyzed sample was over 6 MB.

This technical blog covers information on the Gootkit/AZORult campaign and presents the results of our latest analysis. While we found a few dozen Gootkit samples, we focused on the latest samples (a3c243afceb1fb38f25ae81816891d7d7c11ae76e80a43f31d2ceb9833f2f3df). This variant aimed to steal login accounts from users of five French banks.

Technical Analysis

Gootkit via AZORult

In a previous blog on AZORult, BlackBerry examined the protocol used by the malware to communicate with command-and-control (C2) servers. We implemented a custom scanner to monitor the C2 servers and download and parse configuration settings. The scanner caught some Gootkit campaigns via AZORult occurring on the dates below:

1. Feb. 9 2019 - Feb. 11 2019,
2. Feb. 15 2019 - Mar 7 2019.
3. Mar. 12 2019 - Mar 13 2019.
4. Mar. 31 2019 - Apr. 1 2019.

The threat actor generated Gootkit variants almost every day and used a variety of malware hosting domain names. For more detail of the campaign information, see Table 1 and Table 2:

SHA256 (Gootkit)	Gootkit Delivery URL	Date
879b1f79cbd105ff52ac9a0b01cee4bef921df59e72fb7f5e41db48430ca9a85	hxxp://hairpd[.]com/stat/sputik[.]exe	Feb 9 2019
1d62fec40f14dd0458556f0211529df88d19fe0249d195d6153388610c5525df	hxxp://camdunki[.]com/gx/Aleto[.]exe	Feb 11 2019
b9440e407b971def1d8a3d19c2da0e81145aad4b29e51602fc11f566e9854537	hxxp://camdunki[.]com/gx/wipip[.]exe	Feb 15 2019
d01defb4fe8db26ea0151afa1c4e817db1aba1c8464127110f7ef860164ed79a	hxxp://chermin[.]tweakdsl[.]nl/loges/tlss[.]exe	Feb 16 2019
6b19c0ec581ca24ef2a35d37af523ab2af19740585ad653d98593a057268e01c	hxxp://chermin[.]tweakdsl[.]nl/loges/tlss[.]exe	Feb 17 2019 - Feb 18 2019
2a1dd210ed71e33e58aadd9157eebb35ad38eb70cb182d244a8f9879f195b930	hxxp://chermin[.]tweakdsl[.]nl/loges/tlss[.]exe	Feb 18 2019
1b052c6d721f4dc36b3e58192ac6c664d43aec8f15fcd2a8f91616f705192ebb	hxxp://chermin[.]tweakdsl[.]nl/loges/tlss[.]exe	Feb 19 2019
6a9b222b7be97ed608bcbef6dc05cff9fb16ae9a31a08e719857cad6146dc8d7	hxxp://chermin[.]tweakdsl[.]nl/loges/tlss[.]exe	Feb 20 2019

e67ec7af646f6b9bdfe59e3549f84bd3071c72386de57e9a9a5fe58982dbdd49	hxxp://chermin[.]tweakdsl[.]nl/loges/tlss[.]exe	Feb 21 2019
5eb54a536d9b560b79c7113efa5eebcbe21e9aede3f751d14a98b38c829a53b5	hxxp://chermin[.]tweakdsl[.]nl/loges/zip[.]exe	Feb 22 2019 AM
6c2d6abc2e130092f414d4f64adeb22dac56e8f802d4250c84db62a563bf99ec	hxxp://chermin[.]tweakdsl[.]nl/loges/zip[.]exe	Feb 22 2019 PM
9f6881386d0ff9a0cd2bc49da21b999d13e4ebdc858bdad755ee26898c567a3d	hxxp://chermin[.]tweakdsl[.]nl/loges/zip[.]exe	Feb 23 2019
ab54b89a75ee9d858b734e927ff16aef3d5c8137552c9973864d7eed8aa5e472	hxxp://chermin[.]tweakdsl[.]nl/loges/zip[.]exe	Feb 23 2019 -Feb 24 2019
3974b0985d524dd38a9d040c9eaf880c421411210e0bdd577ac2306f6471a413	hxxp://chermin[.]tweakdsl[.]nl/loges/remove[.]exe	Feb 24 2019 - Feb 25 2019
81a2c4c708c13338ed7aac439aa876c5e1d2116afb23e7cf8a345a05ebd55eed	hxxp://chermin[.]tweakdsl[.]nl/loges/Atrip[.]exe	Feb 26 2019
fb05723ac5960a5776d7432429998d4a48a3f7e74761046352c16712208bd983	hxxp://chermin[.]tweakdsl[.]nl/loges/DlKitlMNdktild[.]exe	Feb 28 2019 AM
983b39b339bc62a09c20ea2f1b1360e17bf9431587e2a257ec4f3a62b4489ff4	hxxp://chermin[.]tweakdsl[.]nl/loges/aHdBhBjUhBHm[.]exe	Feb 28 2019 PM

212aeaf6cec0884743c2c3079dab17eb581dc28329be3e023f62c751cd01169f	hxxp://chermin[.]tweakdsl[.]nl/loges/Astart[.]exe	Mar 1 2019 AM
fcbe6c55b2b092b1d97aa2d8a9ac6f3565b10c47ae7d59b08552d0e2ee11d102	hxxp://chermin[.]tweakdsl[.]nl/loges/Astart[.]exe	Mar 1 2019 PM
729502b7b074e55f1e7d364ae391704376480a28081ca0d7eba4495fca3b1367	hxxp://kbhookah[.]com/loggers/repost[.]exe	Mar 2 2019 - Mar 4 2019
e70a9cfd7c9f1a23d00cdc5eba866ea6c80a4a555498f8d0feba58a765b9aa39	hxxp://kbhookah[.]com/loggers/repost[.]exe	Mar 5 2019
d5ba0f1c01cf12f57cca93996d2f87191c9420afbbd116d3757060d780338d29	hxxp://startintern[.]terweij[.]nl/wp-admin/repox[.]exe	Mar 6 2019
5766bffa91f87cd08582fac05209c5d8d9356ad88e15499038dc624c0ccbc468	hxxp://startintern[.]terweij[.]nl/wp-admin/repox[.]exe	Mar 7 2019
c1ae38afb6c82b9107868d66318095f1c00f1e92dddc0ee953c23a8de4ace353	hxxp://new[.]eltrans53[.]ru/uploads/utf8[.]exe	Mar 12 2019 - Mar 13 2019

Table 1: Gootkit campaign from ssl[.]admin[.]itybuy[.]it (an AZORult C2 server)

SHA256 (Gootkit)	Gootkit Delivery URL	Date

a3c243afceb1fb38f25ae81816891d7d 7c11ae76e80a43f31d2ceb9833f2f3df	hxxp://ccleanerhome[.]com/fonts/igfpers[.]exe	Mar 31 2019 - Apr 1 2019

Table 2: Gootkit campaign from triangularty[.]com (an AZORult C2 server)

Loader and Core DLL Module

Gootkit has two modules: the loader and the core DLL module, as shown in Figure 1 below. The loader is used for evasion, persistence, and downloading the core DLL module. Once the loader downloads the core DLL module, Gootkit can perform malicious actions such as:

- Web injection
- Key logging
- Launching VNC server
- Recording video

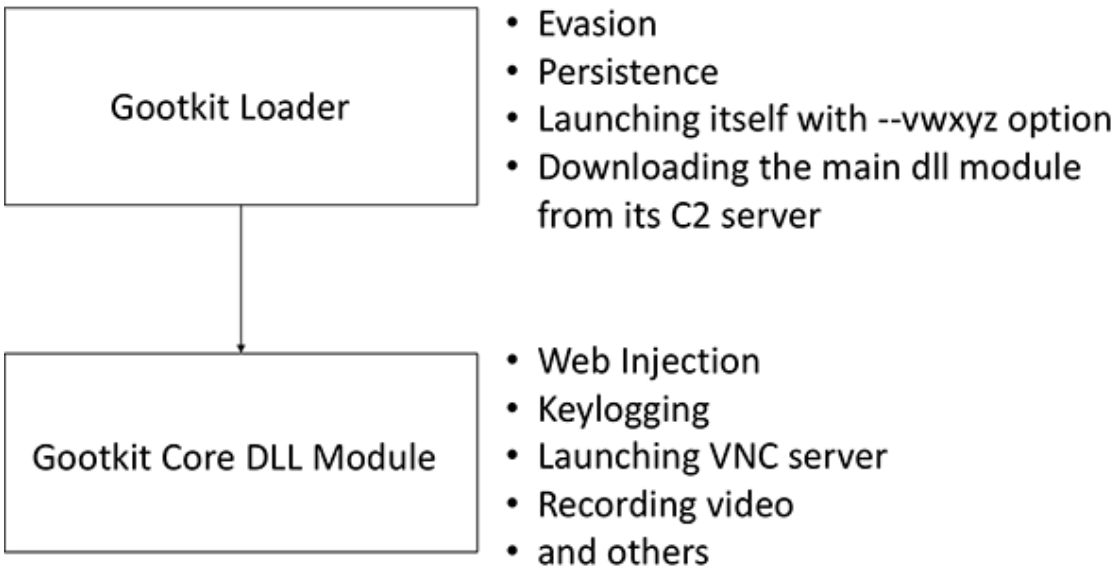


Figure 1: Two modules of Gootkit: Loader and Core DLL Module

Evasion

Gootkit uses many evasion techniques, including anti-VM (virtual machine), anti-debug, and anti-sandbox. It performs the following evasive activities:

1. Check loaded DLL files related to Sandbox technology and debugging by detecting the following strings:
    a. dbghelp.dll (Windows standard debugger's module)
    b. sbiedll.dll (software Sandboxie's module)

2. Check usernames by detecting the following strings:
    a. CurrentUser
    b. Sandbox

3. Check computer names by detecting the following strings.
    a. SANDBOX
    b. 7SILVIA

4. Check registry keys related to the Hardware BIOS information by detecting the strings shown in Table 3:

Registry Key	Registry Entry	Data
HKLM\HARDWARE\DESCRIPTION\System	SystemBiosVersion	• AMI • BOCHS • QEMU • SMCI • INTEL – 6040000 • FTNT-1 • VBOX • SONI
HKLM\HARDWARE\DESCRIPTION\System	VideoBiosVersion	VirtualBox
HKLM\Software\Microsoft\Windows\CurrentVersion	SystemBiosVersion	• 55274-640-2673064-23950 (this value is associated with JoeSandbox) • 76487-644-3177037-23510 (this value is associated with CWSandbox) • 76487-337-8429955-22614 (this value is associated with Anubis)

Table 3: Registry check for evasion

Hardware *SystemBiosVersion* checking code is shown in Figure 2:

```
mov eax,dword ptr ss:[ebp-C]                                    [ebp-C]:"AMI "
xor ebx,ebx
mov edi,dword ptr ss:[ebp-54]                                   [ebp-54]:"FTNT-1"
mov dword ptr ds:[889FE0],eax
mov eax,dword ptr ss:[ebp-74]                                   [ebp-74]:"BOCHS"
mov dword ptr ds:[889FE8],eax
mov eax,dword ptr ss:[ebp-6C]                                   [ebp-6C]:"VBOX"
mov dword ptr ds:[889FEC],eax
mov eax,dword ptr ss:[ebp-68]                                   [ebp-68]:"QEMU"
mov dword ptr ds:[889FF0],eax
mov eax,dword ptr ss:[ebp-70]                                   [ebp-70]:"SMCI"
mov dword ptr ds:[889FF4],eax
mov eax,dword ptr ss:[ebp-64]                                   [ebp-64]:"INTEL  - 6040000"
mov dword ptr ds:[88A000],esi                                   esi:"SONI"
mov esi,ebx                                                     esi:"SONI"
mov dword ptr ds:[889FF8],eax
mov dword ptr ds:[889FFC],edi
push dword ptr ds:[esi+889FE0]
push 169e6ff2db0ed69d283eb34e7161bab8b6dfc72046e510862958de79   889BC8:"INTEL  - 6040000"
call dword ptr ds:[<&StrStrA>]
```

Figure 2: Hardware SystemBiosVersion check for evasion

In addition, the evasive techniques are anti-forensic. Meaningful strings shown in Table 3 are stored in temporary heap memory. At the end of this evasion function, the allocated heap memory is released by using RtlFreeHeap and HeapFree (see Figure 3). The purpose of the evasion is to make memory forensics more difficult to perform:

```
call esi                                    esi:HeapFree
push dword ptr ss:[ebp-B8]                  [ebp-B8]:"SOFTWARE\\Microsoft\\Windows\\CurrentVersion"
xor eax,eax
push eax
call ebx                                    ebx:GetProcessHeap
push eax
call edi                                    edi:RtlFreeHeap
push dword ptr ss:[ebp-7C]                  [ebp-7C]:"VirtualBox"
xor eax,eax
push eax
call ebx                                    ebx:GetProcessHeap
push eax
call esi                                    esi:HeapFree
push dword ptr ss:[ebp-50]                  [ebp-50]:"VideoBiosVersion"
```

Figure 3: Erasing meaningful strings with RtlFreeHeap and HeapFree API

Persistence

Gootkit uses Pending GPO (see Figure 4) to relaunch the malware after reboot. First, it drops an .inf file under the same directory as the Trojan file. The base filename of the INF file is the same as the malware file. For example, if the filename of Gootkit is “igfpers.exe”, the .inf file name is “igfpers.inf”:

```
[Version]
signature = "$CHICAGO$"
AdvancedINF = 2.5, "You need a new version of advpack.dll"

[DefaultInstall]
RunPreSetupCommands = mtwkshrkqooheubzqwbblkwfwtgoivydvjgkoavknp:2

[mtwkshrkqooheubzqwbblkwfwtgoivydvjgkoavknp]
C:\Users\          \Desktop\igfpers.exe
```

Figure 4: An .inf file created by Gootkit

Gootkit then creates three registry values: “Count”, “Path1”, and “Section1” under “HKCU\Software\Microsott\IEAK\GroupPolicy\PendingGPOs” as shown in Figure 5. “Path1” contains the full path of the .inf file and “Section1” refers to a section name (“DefaultInstall”) written in the .inf file:

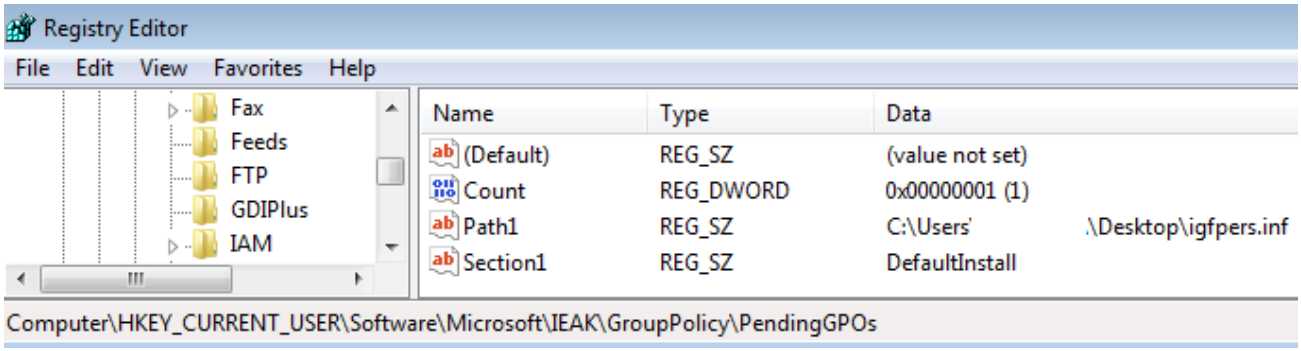


Figure 5: Registry values for its persistence

Core DLL Module Installation

Gootkit loader launches itself using the “--vwxyz” options. It then downloads an encoded DLL module from the C2 server and injects it into a Gootkit process. The C2 server will not send the DLL module without the appropriate “UserAgent”. Once it receives a valid response from the C2 server, Gootkit splits the encoded DLL module into chunks. Each chunk size is 512,000 bytes (at most) and is saved under “HKCU\Software\AppData\Low\finget_{index number}” (See Figure 6). After reboot, Gootkit loads the data from registry keys and concatenates all chunks into an encoded DLL. This results in the core DLL module becoming fileless:

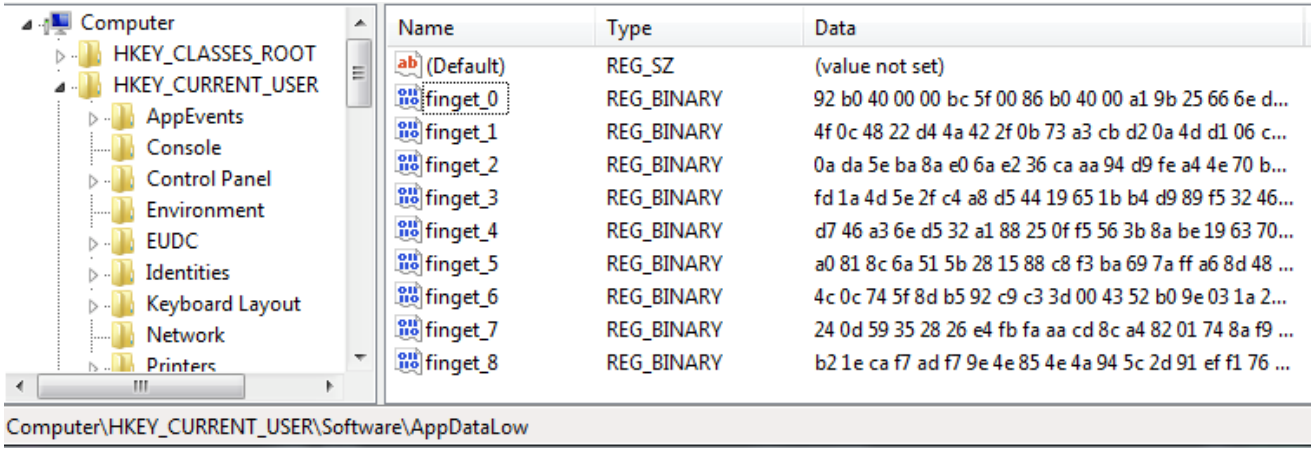


Figure 6: Registry values for its persistence

The DLL module is decrypted and decompressed by “RTLDecompressBuffer”. Gootkit then allocates a new memory section to the current process and copies it into the allocated memory.

The DLL module contains JavaScript files for performing malicious activity. During our investigation, we found over a hundred embedded JS files. Most of them are innocent (Node.js library). However, some JavaScript files are intended for malicious purposes such as

“malware.js”, “spyware.js”, “zeusmask.js”, and so on. Malicious JavaScript codes are responsible for backdoor functions which allow Gootkit to:

- Update the DLL
- Launch VNC server
- Capture keystrokes
- Inject malicious scripts for stealing online banking credentials
- Record video
- Steal email
- Perform other malicious actions

In addition, a JavaScript file is designed to detect VM environments (Figure 7):

```
function IsVirtualMachine() {

    //print('IsVirtualMachine >>> ');
    var bIsVirtualMachine = false;

    try{
        var VMBioses = [
            "AMI ",
            "BOCHS",
            "VBOX",
            "QEMU",
            "SMCI",
            "INTEL  - 6040000",
            "FTNT-1",
            "SONI"
        ];


        var SystemBiosVersion =
            (new reg.WindowsRegistry(HKEY_LOCAL_MACHINE, "HARDWARE\\DESCRIPTION\\System", KEY_READ, true)
                .ReadString("SystemBiosVersion") || "hui").toString();


        for (let i = 0; i < VMBioses.length; i++) {
            if (SystemBiosVersion.toLowerCase().indexOf(VMBioses[i].toLowerCase()) !== -1) {
                bIsVirtualMachine = true;
                break;
            }
        }

        var ideDevices = VmCheckGetDisksArray('SYSTEM\\CurrentControlSet\\Enum\\IDE');
        var scsiDevices = VmCheckGetDisksArray('SYSTEM\\CurrentControlSet\\Enum\\SCSI');
```

Figure 7: VM check

Banking Trojan

The DLL module receives web injection code from its C2 server and tries to steal login accounts from victims who used five French banks. Figure 9 shows a code snippet of web injection script. The threat monitors the victim’s web browser and steals credentials when the French banks are accessed. The link at the bottom of Figure 8 stored a French-bank-specific script:

Figure 8: A web injection script against a French bank

A JavaScript code targeting one French bank was also able to steal PIN numbers.

Cautious users might choose to use a software keyboard to input sensitive information on online banking websites. However, Gootkit displays a fake software keyboard designed to steal user input (and the victim's PIN number).

Conclusion

This blog covered a Gootkit campaign that spread using AZORult infostealer. During this campaign, attackers constantly changed the hash values of Gootkit and the hosting URLs. Based upon our monitoring, the campaign was active between February and April 2019. The sample we analyzed was aimed at stealing banking information from users of five French banks. Gootkit is still being actively used against victims in EU regions.

If you are using BlackBerry's endpoint protection solution CylancePROTECT®, you are proactively protected from Gootkit. Blackberry uses artificial intelligence-based agents trained for threat detection on millions of both safe and unsafe files. Our automated security agents block Gootkit based on countless file attributes and malicious behaviors instead of relying on a specific file signature. CylancePROTECT, which offers a predictive advantage over zero-day threats, is trained on and effective against both new and legacy cyberattacks.

For more information visit https://www.cylance.com and https://www.blackberry.com/us/en.

Appendix

Indicators of Compromise (IOCs)

- **Hashes**
  - **Gootkit loader:**
    - a3c243afceb1fb38f25ae81816891d7d7c11ae76e80a43f31d2ceb9833f2f3df
    - c1ae38afb6c82b9107868d66318095f1c00f1e92dddc0ee953c23a8de4ace353
    - 5766bffa91f87cd08582fac05209c5d8d9356ad88e15499038dc624c0ccbc468
    - d5ba0f1c01cf12f57cca93996d2f87191c9420afbbd116d3757060d780338d29
    - e70a9cfd7c9f1a23d00cdc5eba866ea6c80a4a555498f8d0feba58a765b9aa39
    - 729502b7b074e55f1e7d364ae391704376480a28081ca0d7eba4495fca3b1367
    - fcbe6c55b2b092b1d97aa2d8a9ac6f3565b10c47ae7d59b08552d0e2ee11d102
    - 212aeaf6cec0884743c2c3079dab17eb581dc28329be3e023f62c751cd01169f
    - 983b39b339bc62a09c20ea2f1b1360e17bf9431587e2a257ec4f3a62b4489ff4
    - fb05723ac5960a5776d7432429998d4a48a3f7e74761046352c16712208bd983
    - 81a2c4c708c13338ed7aac439aa876c5e1d2116afb23e7cf8a345a05ebd55eed
    - 3974b0985d524dd38a9d040c9eaf880c421411210e0bdd577ac2306f6471a413
    - ab54b89a75ee9d858b734e927ff16aef3d5c8137552c9973864d7eed8aa5e472
    - 9f6881386d0ff9a0cd2bc49da21b999d13e4ebdc858bdad755ee26898c567a3d
    - 6c2d6abc2e130092f414d4f64adeb22dac56e8f802d4250c84db62a563bf99ec
    - 5eb54a536d9b560b79c7113efa5eebcbe21e9aede3f751d14a98b38c829a53b5
    - e67ec7af646f6b9bdfe59e3549f84bd3071c72386de57e9a9a5fe58982dbdd49
    - 6a9b222b7be97ed608bcbef6dc05cff9fb16ae9a31a08e719857cad6146dc8d7
    - 1b052c6d721f4dc36b3e58192ac6c664d43aec8f15fcd2a8f91616f705192ebb
    - 2a1dd210ed71e33e58aadd9157eebb35ad38eb70cb182d244a8f9879f195b930
    - 6b19c0ec581ca24ef2a35d37af523ab2af19740585ad653d98593a057268e01c
    - d01defb4fe8db26ea0151afa1c4e817db1aba1c8464127110f7ef860164ed79a
    - b9440e407b971def1d8a3d19c2da0e81145aad4b29e51602fc11f566e9854537
    - 1d62fec40f14dd0458556f0211529df88d19fe0249d195d6153388610c5525df
    - 879b1f79cbd105ff52ac9a0b01cee4bef921df59e72fb7f5e41db48430ca9a85

- **Domain names**
  - Gootkit’s C2 server:
    - sillikogermin[.]com
    - feferturietan[.]com
    - manjuorlidnqo[.]com
    - chechelderpos[.]com
    - kalamindridro[.]com
    - avant-garde[.]host
    - kinzhal[.]online
    - servicemanager[.]icu
    - partnerservice[.]xyz
    - Location of web injection script
    - mabanquesecure[.]com

- **URLs**
  - Location of Gootkit DLL
    sillikogermin[.]com/rbody32

- **C2s/IPs**
  - Gootkit delivery URL
    - hxxp://ccleanerhome[.]com/fonts/igfpers[.]exe
    - hxxp://new[.]eltrans53[.]ru/uploads/utf8[.]exe
    - hxxp://startintern[.]terweij[.]nl/wp-admin/repox[.]exe
    - hxxp://kbhookah[.]com/loggers/repost[.]exe
    - hxxp://chermin[.]tweakdsl[.]nl/loges/Astart[.]exe
    - hxxp://chermin[.]tweakdsl[.]nl/loges/aHdBhBjUhBHm[.]exe
    - hxxp://chermin[.]tweakdsl[.]nl/loges/DlKitlMNdktild[.]exe
    - hxxp://chermin[.]tweakdsl[.]nl/loges/Atrip[.]exe
    - hxxp://chermin[.]tweakdsl[.]nl/loges/remove[.]exe
    - hxxp://chermin[.]tweakdsl[.]nl/loges/zip[.]exe
    - hxxp://chermin[.]tweakdsl[.]nl/loges/tlss[.]exe
    - hxxp://camdunki[.]com/gx/wipip[.]exe
    - hxxp://camdunki[.]com/gx/Aleto[.]exe
    - hxxp://hairpd[.]com/stat/sputik[.]exe

- **Persistence**
  - .INI file
    - e.g., *C:\Users\USERNAME\Desktop\igfpers.inf*
  - Registry Key
    - HKCU\Software\Microsott\IEAK\GroupPolicy\PendingGPOs
  - Registry Value / Data
    - Path1 / *Path to Gootkit loader*
    - Section1 / DefaultInstall (It points to a section in .inf file)

- **Interesting Strings**
  - --vwxyz (An option for new process creation to download Gootkit DLL from its C2 server)

About Masaki Kasuya

Senior Threat Researcher at BlackBerry Cylance, Japan

Masaki Kasaki started his professional career as Security Engineer at a large e-commerce company and earned practical experience in malware analysis, penetration testing, incident response, and corporate IT security. His Ph.D. dissertation sought how to stimulate stealthy malware's behavior. While he was Ph.D. student, he received student paper award and student presentation award. He holds SANS GREM, GCFA, GCIH, GCIA and GMOB.

About Tatsuya Hasegawa

Senior Threat Researcher at BlackBerry Cylance

Tatsuya Hasegawa is a Senior Threat Researcher in APAC at BlackBerry, and is responsible for malware analysis and sandbox technology. He has practical experience in the both managed security service provider as a security analyst and CSIRT as an incident handler. His certifications include: GREM, GCIH, GCFA, GXPN, GPEN and CISSP.

Back