Winnti Abuses GitHub for C&C Communications

By By: Trend Micro Mar 22, 2017 Read time: 5 min (1226 words)

Published: 2017-03-22 · Archived: 2026-04-05 12:45:41 UTC

With additional analysis from Cyber Safety Solutions Team

Developers constantly need to modify and rework their source codes when releasing new versions of applications or coding projects they create and maintain. This is what makes GitHub—an online repository hosting service that provides version control management—popular. In many ways, it's like a social networking site for programmers and developers, one that provides a valuable platform for code management, sharing, collaboration, and integration.

GitHub is no stranger to misuse, however. Open-source ransomware projects EDA2 and Hidden Tear—supposedly created for educational purposes—were hosted on GitHub, and have since spawned various offshoots that have been found targeting enterprises. Tools that exploited vulnerabilities in Internet of Things (IoT) devicesopen on a new tab were also made available on GitHub. Even the Limitless Keyloggeropen on a new tab, which was used in targeted attacks, was linked to a GitHub project.

Recently, the Winnti group, a threat actor with a past of traditional cybercrime -particularly with financial fraud, has been seen abusing GitHub by turning it into a conduit for the command and control (C&C) communications of their seemingly new backdoor (detected by Trend Micro as BKDR64_WINNTI.ONM).

Our research also showed that the group still uses some of the infamous PlugX malware variants—a staple in Winnti's arsenal—to handle targeted attack operations via the GitHub account we identified.

Malware Analysis

The malware we analyzed is separated in two files: a loader, and the payload.

The loader, named *loadperf.dll*, is a modified version of its legitimate, similarly named counterpart—a Microsoft file which helps manipulate the performance registry. An extra component has been added to its sections. It copies itself on *%WINDIR%\system32\wbem* and replaces the original DLL. It leverages the WMI performance adapter service (wmiAPSrv), a legitimate file in Windows that collects information related to system performance, to import the loader via *services.exe*. The system also imports all related DLL files and includes the payload "loadoerf.ini". The infection chain includes an additional (albeit empty) function imported from *loadoerf.ini*, *gzwrite64*, which works as a fake Application Program Interface (API) that serves as the payload's entry point. Although *gzwrite64* is imported by *loadperf.dll*, the payload's main function is actually located in the *DLLMain* of "loadoerf.ini".


Figure 1: Extra section .idata added to the original loadperf.dll


Figure 2: Extra imported function gzwrite64

The payload is a file named *loadoerf.ini* that contains decryption, run, and code injection functions. When it is loaded by the system, *DLLMain* decrypts the payload via *CryptUnprotectData*. Since the function highly depends on the actual "machine ID", decryption on another machine that isn't the original infected host is not viable, making malware analysis more difficult.

 *Figure 3: Part of the decryption function used in the payload*

After decryption, partial code is run on the machine, which is then injected to *svchost.exe* (a key Windows component); payload is then loaded into memory.

 *Figure 4: Execution/infection flow of loadoerf.ini*

How is GitHub abused? Upon successful infection, the malware starts communicating with an HTML page from a repository stored in a GitHub project.

 *Figure 5: GitHub account hosting an HTML page used for C&C communication*

Any malware threat analyst will immediately recognize Line 3 in the image above as a potential PlugX-encrypted line. The beginning and end markers, DZKS and DZJS, are typicalopen on a new tab in PlugX. A closer look, however, shows that the decryption algorithm is different from PlugX. In this case, decrypting them reveals references to its actual command and control (C&C) server: an IP address and a port number the malware will connect to.

Winnti currently uses different encryption algorithms to store those C&C references in the files they stored on Github. Among them is an algorithm utilized by PlugX. In fact, we found references to PlugX in the C&C strings we analyzed, indicating that the group may also be using the same backdoor in this particular campaign. Although we were unable to find a PlugX sample through that particular GitHub, we surmise some PlugX variants in the wild use this GitHub repository to get their C&C information.

Nearly all the other algorithms used in this GitHub campaign are derived from the original PlugX algorithm:

- PlugX style + shift string + Base64
- PlugX style + shift string + Base64 + XOR
- PlugX style + Base64 + XOR

One algorithm is also built in mark strings + shift string + Base64 encoding.

Following Winnti's Trails

The GitHub account used by the threat actor was created in May 2016. It created one legitimate project/repository (*mobile-phone-project*) in June 2016, derived from another generic GitHub page.

The repository for Winnti's C&C communications was created on August 2016. We surmise that the GitHub account was not compromised, and instead created by Winnti. By March 2017, the repository already contained 14 different HTML pages created at various times.

Timeline of the Campaign

We mapped Winnti's activities for this campaign by analyzing the dates exposed in GitHub. For each file, GitHub stores first-and-last commit timestamps; these enabled us to create a timeline of the first use of the group's many C&C servers.

We monitored the period during which IP addresses were found connecting to Winnti's C&C servers and found that they started their operations in the afternoon up to late evening. The timetable resembles traditional working hours for cybercriminals, compared to those with less structure who prefer starting their days late, but also working until very late hours. In fact, we only observed one instance of activity during the weekend, where a new HTML file was created.

The earliest activity we tracked on the GitHub account was from August 17, 2016, with the most recent in March 12, 2017.

Here is a timeline of when the C&C server's IP addresses were first used, based on our monitoring:

 *Figure 6: Timeline of the C&C server's IP addresses*

C&C Servers

The GitHub account used by Winnti shows 12 different IP addresses, with various port numbers used for them. All communication to these C&C servers are done on three different port numbers: 53 (DNS), 80 (HTTP), and 443 (HTTPS). These are typical techniques PlugX and Winnti malware variants use to communicate between compromised machines and their C&C servers. Nearly all the C&C servers are hosted in the U.S., while two are located in Japan.

C&C Server's IP Address	**Port Number**
160[.]16[.]243[.]129	443 (HTTPS)
160[.]16[.]243[.]129	53 (DNS)
160[.]16[.]243[.]129	80 (HTTP)
174[.]139[.]203[.]18	443 (HTTPS)
174[.]139[.]203[.]18	53 (DNS)
174[.]139[.]203[.]20	53 (DNS)
174[.]139[.]203[.]22	443 (HTTPS)
174[.]139[.]203[.]22	53 (DNS)
174[.]139[.]203[.]27	53 (DNS)
174[.]139[.]203[.]34	53 (DNS)
174[.]139[.]62[.]58	80 (HTTP)

174[.]139[.]62[.]60	443 (HTTPS)
174[.]139[.]62[.]60	53 (DNS)
174[.]139[.]62[.]60	80 (HTTP)
174[.]139[.]62[.]61	443 (HTTPS)
61[.]195[.]98[.]245	443 (HTTPS)
61[.]195[.]98[.]245	53 (DNS)
61[.]195[.]98[.]245	80 (HTTP)
67[.]198[.]161[.]250	443 (HTTPS)
67[.]198[.]161[.]250	53 (DNS)
67[.]198[.]161[.]251	443 (HTTPS)
67[.]198[.]161[.]252	443 (HTTPS)

Figure 6: IP addresses used for C&C communication, and the port numbers they use

We have privately disclosed our findings to GitHub prior to this publication and are proactively working with them about this threat.

Conclusion

Abusing popular platforms like GitHub enables threat actors like Winnti to maintain network persistence between compromised computers and their servers, while staying under the radar. Although Winnti may still be employing traditional malware, its use of a relatively unique tactic to stay ahead of the threat landscape's curve reflects the increased sophistication that threat actors are projected to employpredictions.

Related Hashes (SHA256) detected as BKDR64_WINNTI.ONM:

06b077e31a6f339c4f3b1f61ba9a6a6ba827afe52ed5bed6a6bf56bf18a279ba — *cryptbase.dll*
1e63a7186886deea6c4e5c2a329eab76a60be3a65bca1ba9ed6e71f9a46b7e9d – *loadperf.dll*
7c37ebb96c54d5d8ea232951ccf56cb1d029facdd6b730f80ca2ad566f6c5d9b – *loadoerf.ini*
9d04ef8708cf030b9688bf3e8287c1790023a76374e43bd332178e212420f9fb — *wbemcomn.ini*
b1a0d0508ee932bbf91625330d2136f33344ed70cb25f7e64be0620d32c4b9e2 — *cryptbase.ini*
e5273b72c853f12b77a11e9c08ae6432fabbb32238ac487af2fb959a6cc26089 — *wbemcomn.dll*
