STOMP 2 DIS: Brilliance in the (Visual) Basics

fireeye.com/blog/threat-research/2020/01/stomp-2-dis-brilliance-in-the-visual-basics.html

Threat Research

Rick Cole, Andrew Moore, Genevieve Stark, Blaine Stancill

Feb 05, 2020

15 mins read

Malware

Threat Research

Throughout January 2020, FireEye has continued to observe multiple targeted phishing campaigns designed to download and deploy a backdoor we track as MINEBRIDGE. The campaigns primarily targeted financial services organizations in the United States, though targeting is likely more widespread than those we've initially observed in our FireEye product telemetry. At least one campaign targeted South Korean organizations, including a marketing agency.

In these campaigns, the phishing documents appeared to be carefully crafted and leveraged some publicly-documented — but in our experience uncommon and misunderstood — TTPs, likely in an effort to decrease detection of the malicious documents' macros. The actor also used a self-hosted email marketing solution across multiple campaigns. Notably, the payload delivered in these campaigns leveraged a packer previously affiliated with a commonly-tracked threat actor, an overlap that we will explore later.

This blog post will review the theme of these campaigns and their targets, the adversary's unique tradecraft, the MINEBRIDGE C++ backdoor, some potential attribution overlaps, and importantly — the threat actor's love of rap music.

Targeting and Lure Detail

While we first identified MINEBRIDGE samples in December, we observed our first phishing campaigns relating to this activity in early January 2020. Email addresses used to send phishing messages were associated with domains that appear to have been registered specifically for this purpose within a few weeks of the activity — and were thematically consistent with the content of the phishing messages.

Additionally, the actor(s) responsible are likely using a self-hosted email marketing solution called Acelle. Acelle adds extended email headers to messages sent via the platform in the format of X-Acelle-<variable>. The messages observed across campaigns using these TTPs have included a "Customer-Id" value matching "X-Acelle-Customer-Id: 5df38b8fd5b58". While that field remained consistent across all observed campaigns, individual campaigns also shared overlapping "X-Acelle-Sending-Server_Id" and "X-Acelle-Campaign-Id" values. All of the messages also included a "List-Unsubscribe" header offering a link hosted at 45.153.184.84 suggesting that it is the server hosting the Acelle instance used across these campaigns. The sample table for one campaign below illustrates this data:

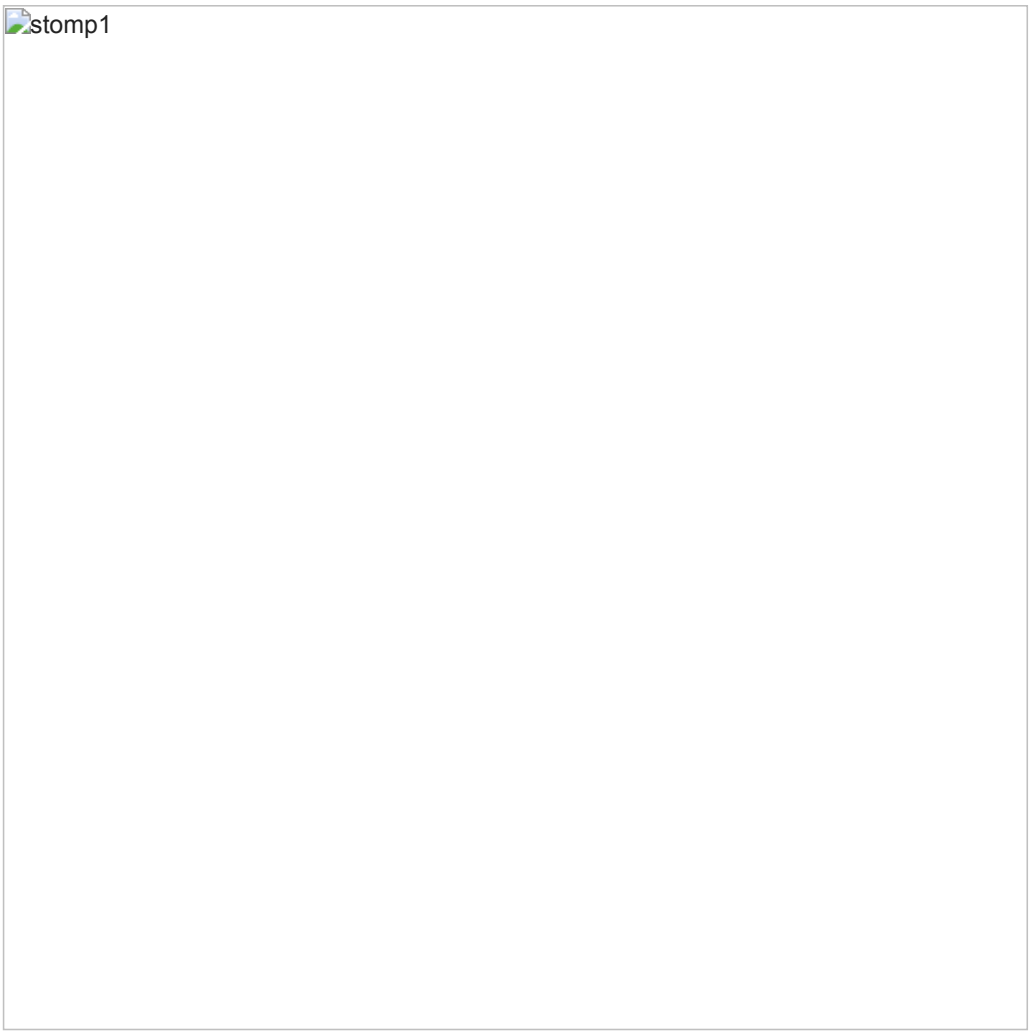
Timestamp	Sender	Subject	x-acelle-subscriber-id	x-acelle-sending-server-id	x-acelle-customer-id	x-acelle-campaign-id
1/7/20 16:15	info@rogervecpa.com	tax return file	25474792e6f8c	5e14a2664ffb4	5df38b8fd5b58	5e14a2664ffb4
1/7/20 15:59	info@rogervecpa.com	tax return file	22e183805a051	5e14a2664ffb4	5df38b8fd5b58	5e14a2664ffb4
1/7/20	info@rogervecpa.com	tax return file	657e1a485ed77	5e14a2664ffb4	5df38b8fd5b58	5e14a2664ffb4
1/7/20 16:05	info@rogervecpa.com	tax return file	ddbbffbcb5c6c	5e14a2664ffb4	5df38b8fd5b58	5e14a2664ffb4

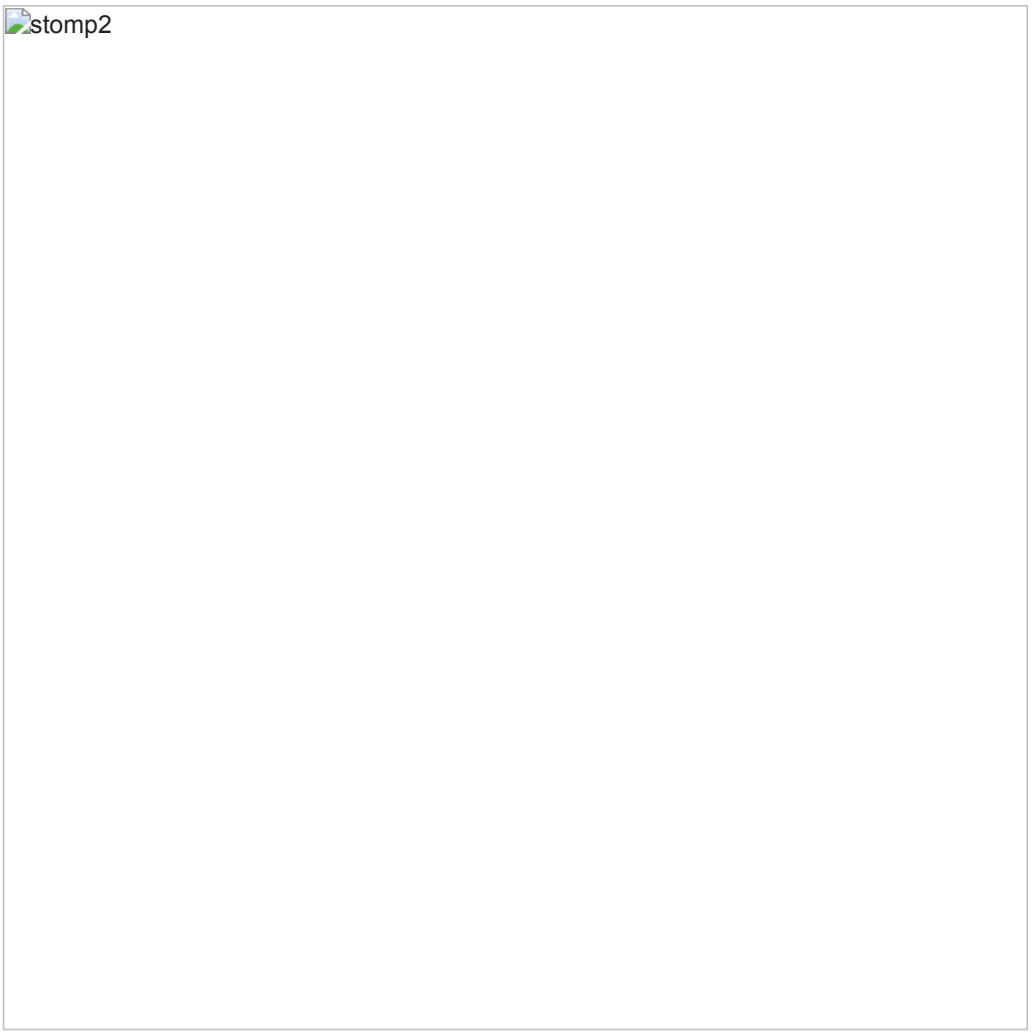
The URLs requested by the malicious documents and serving the final MINEBRIDGE payloads delivered in each of these campaigns provide additional overlap across campaigns. In all observed cases, the domains used the same bullet-proof hosting service. The URI used to download the final payload was "/team/invest.php" or, in one case, "/team/rumba.php". Perhaps the most fun overlap, however, was discovered when trying to identify additional artifacts of interest hosted at similar locations. In most cases a GET request to the parent directory of "/team/" on each of the identified domains served up the lyrics to rap group Onyx's "Bang 2 Dis" masterpiece. We will refrain from sharing the specific verse hosted due to explicit content.

One of the more notable characteristics of this activity was the consistency in themes used for domain registration, lure content, similarities in malicious document macro content, and targeting. Since first seeing these emails, we've identified at least 3 distinct campaigns.

Campaign #1: January 7, 2020 – Tax Theme

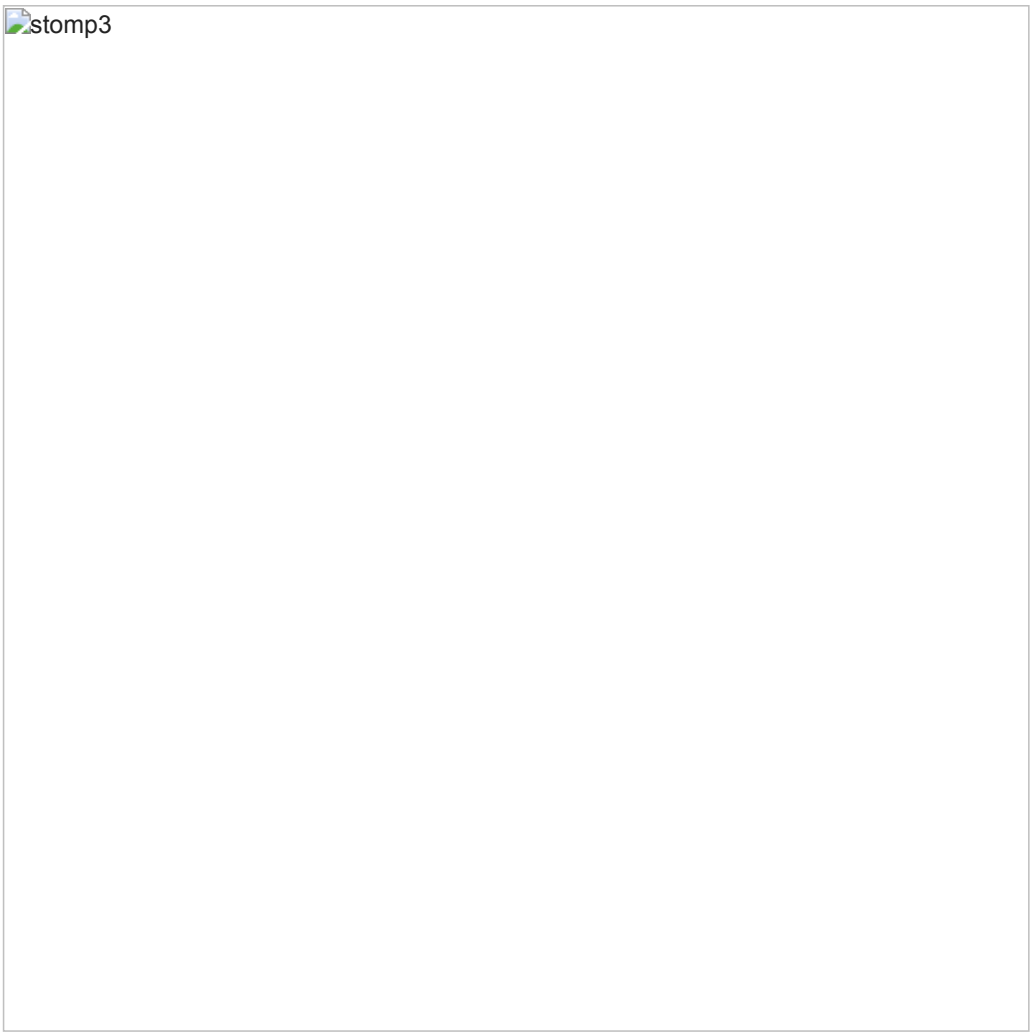
- Emails associated with this campaign used the CPA themed domain rogervecpa.com registered in late November and the subject line "Tax Return File" with IRS related text in the message body.
- The attached payload was crafted to look like an H&R Block related tax form.
- Observed targeting included the financial sector exclusively.

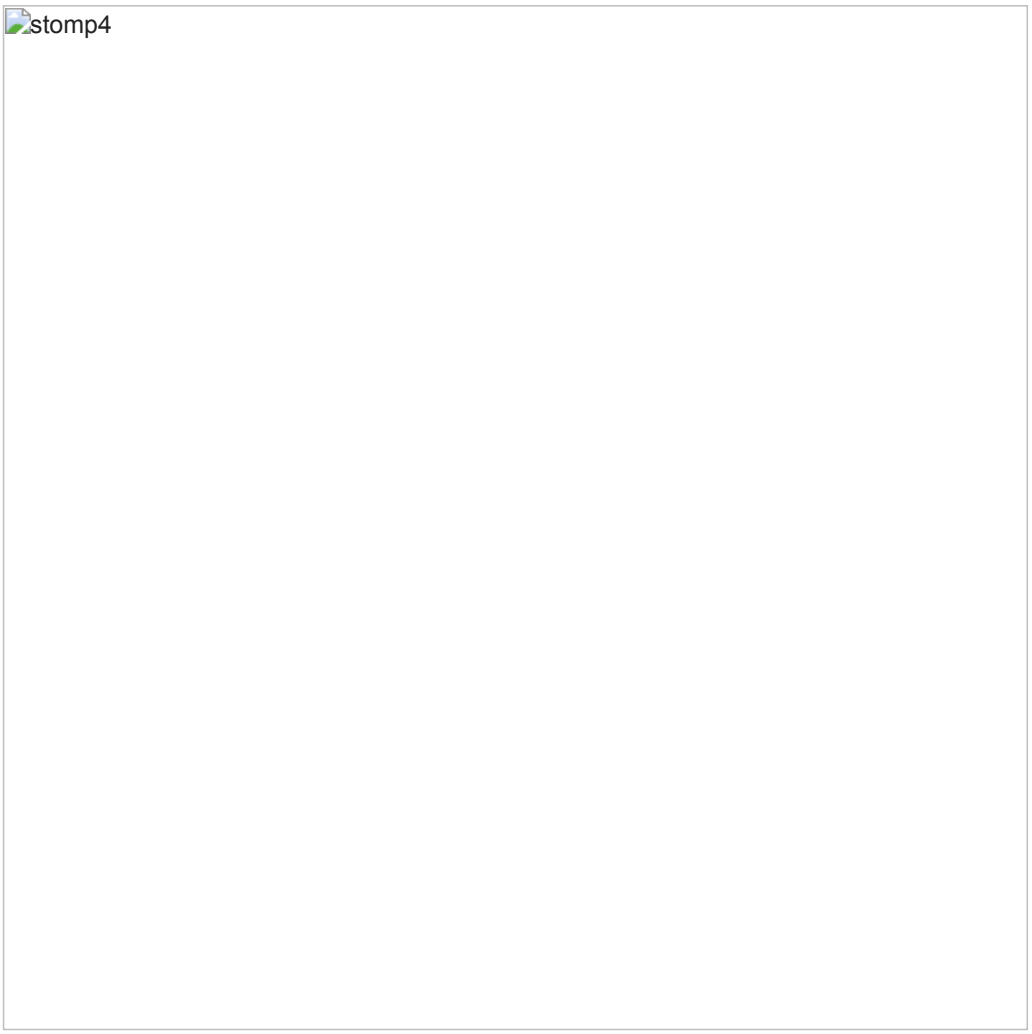

stomp1



Campaign #2: January 8, 2020 – Marketing Theme

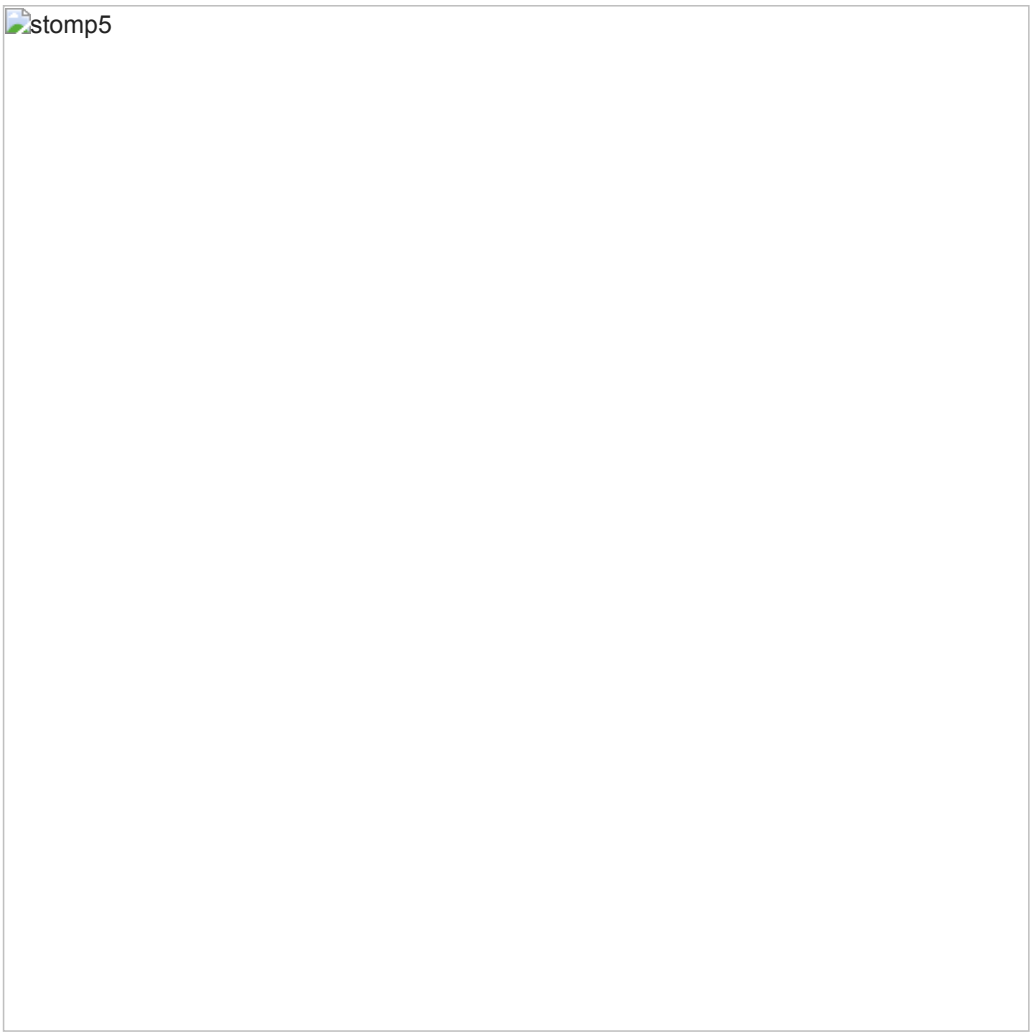
- Emails associated with this campaign used the same CPA themed domain rogervecpa.com along with pt-cpaaccountant.com, also registered late November.
- The subject line and message body offered a marketing partnership opportunity to the victim.
- The attached payload used a generic theme enticing users to enable macro content.
- Observed targeting focused on a South Korean marketing agency.

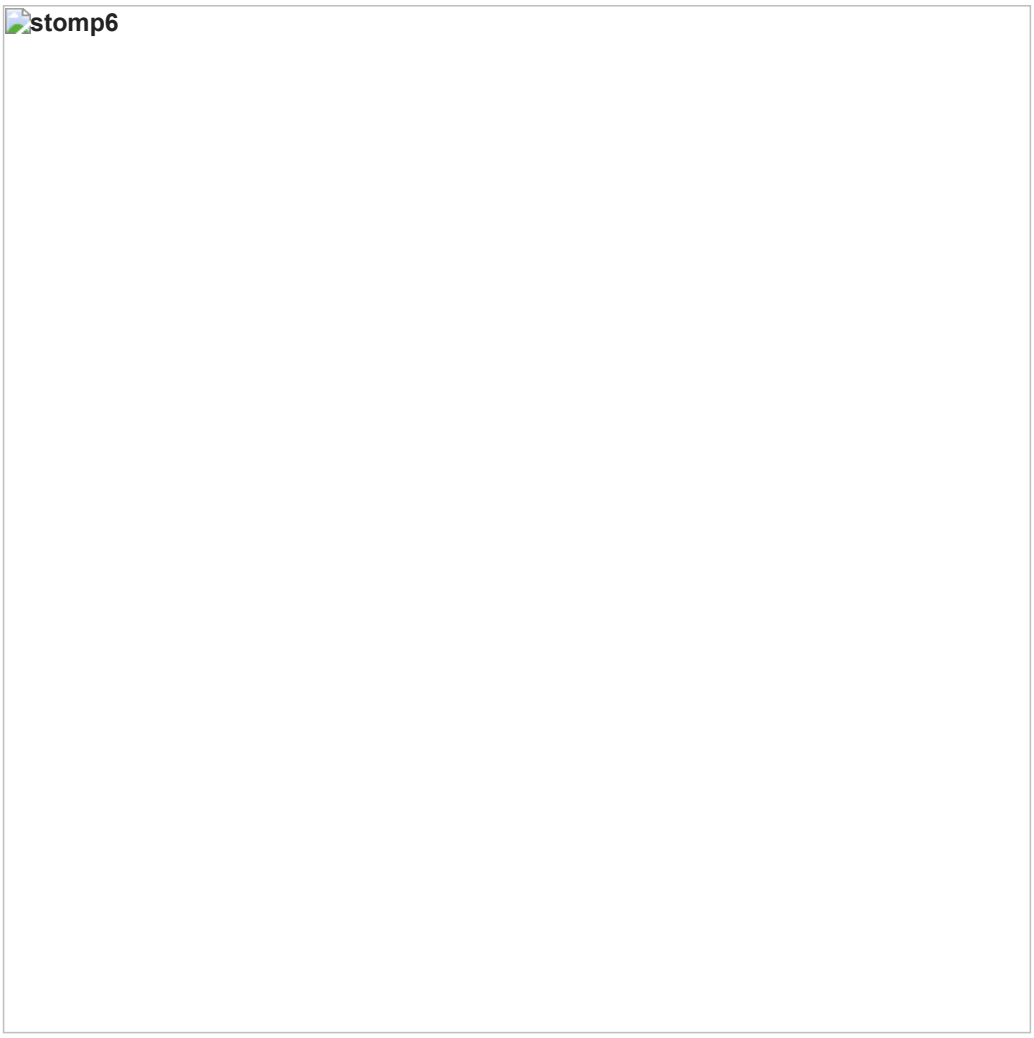

stomp3



Campaign #3: January 28, 2020 – Recruiting Theme

- Emails associated with this campaign were sent from several different email addresses, though all used the recruiting-themed domain agent4career.com which was registered on January 20, 2020.
- The subject line and message body referenced an employment candidate with experience in the financial sector.
- The attached payload masqueraded as the resume of the same financial services candidate referenced in the phishing email.
- Observed targeting included the financial sector exclusively.


stomp5



Quit Stepping All Over My Macros

The phishing documents themselves leverage numerous interesting TTPs including hiding macros from the Office GUI, and VBA stomping.

VBA stomping is a colloquial term applied to the manipulation of Office documents where the source code of a macro is made to mismatch the pseudo-code (hereto referred to as "p-code") of the document. In order to avoid duplicating research and wasting the reader's time, we will instead reference the impressive work of our predecessors and peers in the industry. As an introduction to the concept, we first recommend reading the tool release blog post for EvilClippy from Outflank. The security team at Walmart has also published incredible research on the methodology. Vesselin Bontchev provides a useful open source utility for dumping the p-code from an Office document in pcodedmp. This tool can be leveraged to inspect the p-code of a document separate from its VBA source. It was adopted by the wider open source analysis toolkit oletools in order to detect the presence of stomping via comparison of p-code mnemonics vs keyword extraction in VBA source.

That is a whole lot of quality reading for those interested. For the sake of brevity, the most important result of VBA stomping as relevant to this blog post is the following:

- Static analysis tools focusing on VBA macro source extraction may be fooled into a benign assessment of a document bearing malicious p-code.
- When VBA source is removed, and a document is opened in a version of Office for which the p-code was not compiled to execute, a macro will not execute correctly, resulting in potential failed dynamic analysis.
- When a document is opened under a version of Office that uses a VBA version that does not match the version of Office used to create the document, VBA source code is recompiled back into p-code.

- When a document is opened in Office and the GUI is used to view the macro, the embedded p-code is decompiled to be viewed.

The final two points identify some interesting complications in regard to leveraging this methodology more broadly. Versioning complexities arise that toolkits like EvilClippy leverage Office version enumeration features to address. An actor's VBA stomped document containing benign VBA source but evil p-code must know the version of Office to build the p-code for, or their sample will not detonate properly. Additionally, if an actor sends a stomped document, and a user or researcher opens the macro in the Office editor, they will see malicious code.

Our actor addressed the latter point of this complication by leveraging what we assess to be another feature of the EvilClippy utility, wherein viewing the macro source is made inaccessible to a user within Office by modifying the PROJECT stream of the document. Let's highlight this below using a publicly available sample we attribute to our actors (SHA256: 18698c5a6ff96d21e7ca634a608f01a414ef6fbbd7c1b3bf0f2085c85374516e):

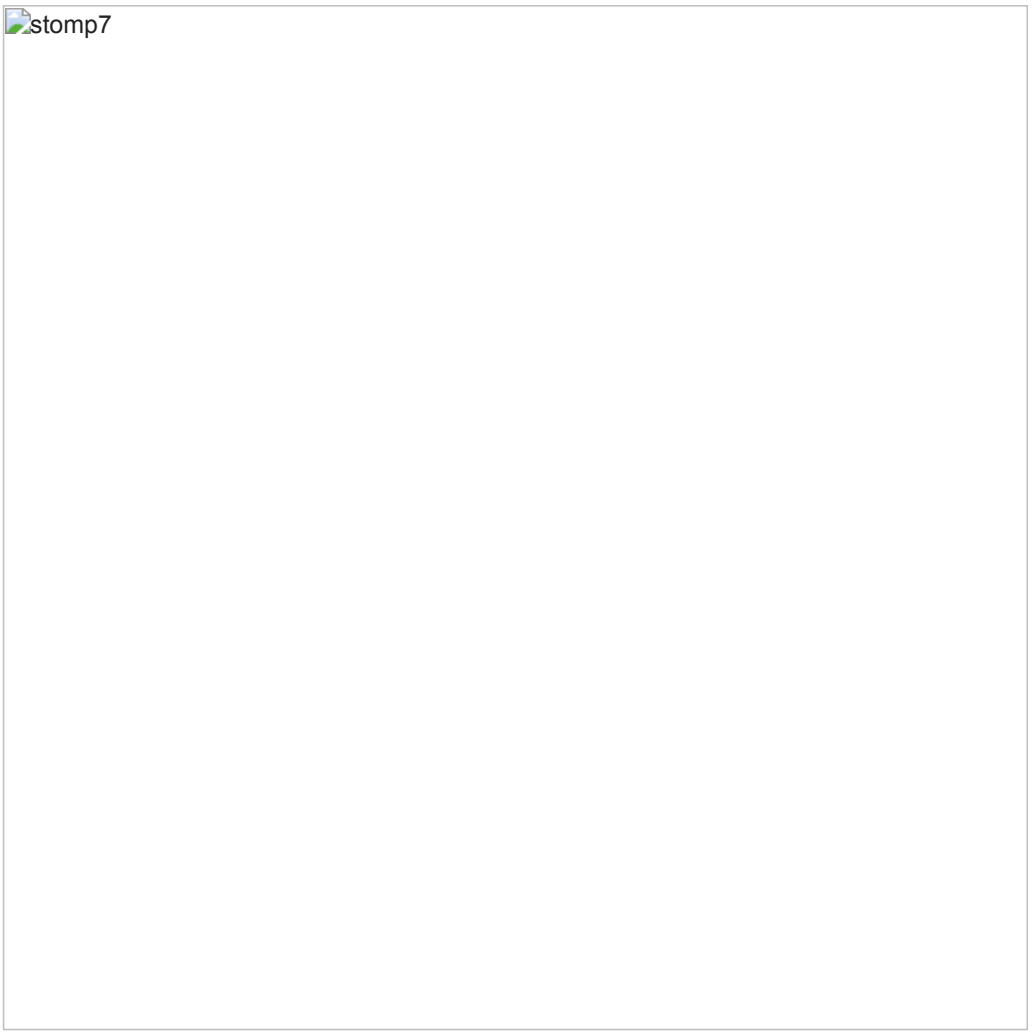
Document PROJECT stream:

```
ID="{33C06E73-23C4-4174-9F9A-BA0E40E57E3F}"
Document=ThisDocument/&H00000000
Name="Project"
HelpContextID="0"
VersionCompatible32="393222000"
CMG="A3A1799F59A359A359A359A3"
DPB="87855DBBA57B887C887C88"
GC="6B69B1A794A894A86B"
[Host Extender Info]
&H00000001={3832D640-CF90-11CF-8E43-00A0C911005A};VBE;&H00000000
[Workspace]
ThisDocument=0, 0, 0, 0, C
Module1=26, 26, 388, 131, Z
```

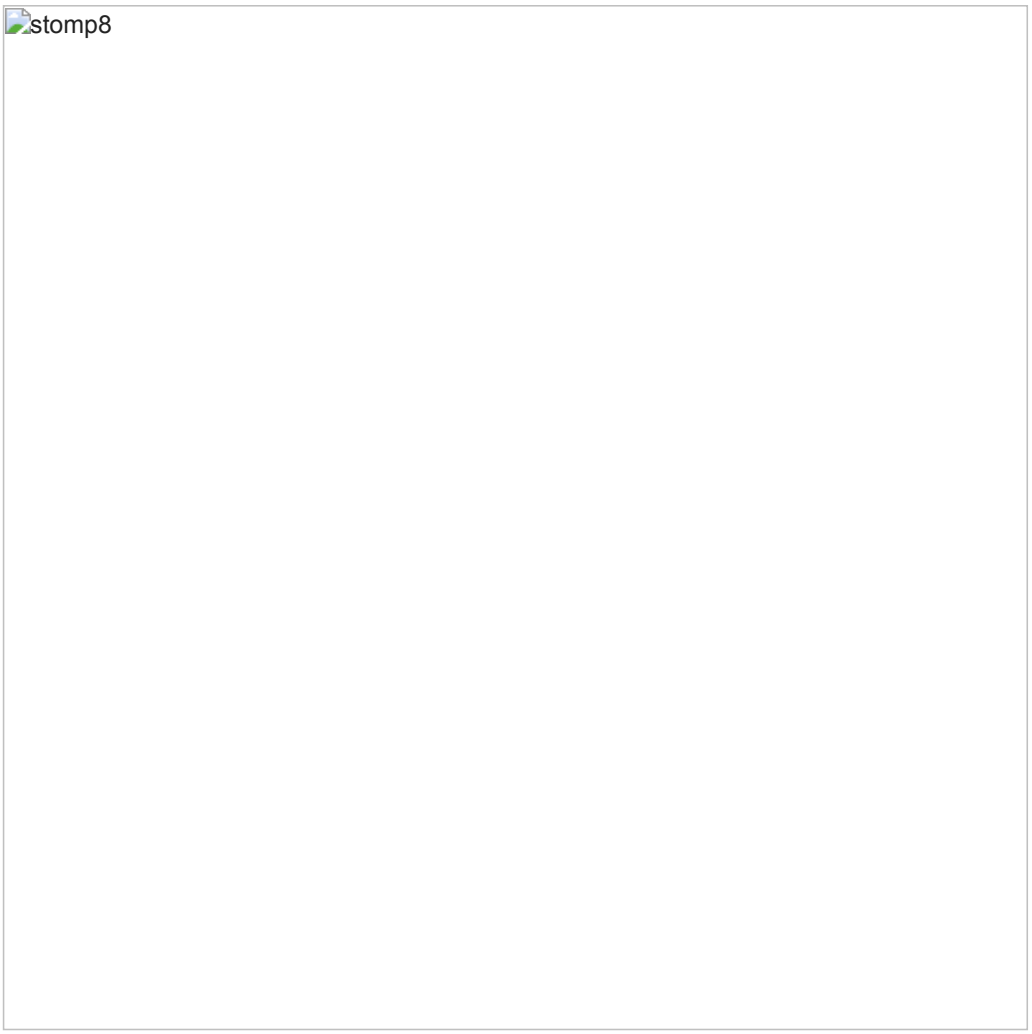
The above PROJECT stream has been modified. Within the PROJECT stream workspace, a module is referenced. However, there is no module defined. We would expect the unmodified PROJECT stream of this document prior to utilization of a tool to modify it to be as follows:

```
ID="{33C06E73-23C4-4174-9F9A-BA0E40E57E3F}"
Document=ThisDocument/&H00000000
Module=”Module1”
Name="Project"
HelpContextID="0"
VersionCompatible32="393222000"
CMG="A3A1799F59A359A359A359A3"
DPB="87855DBBA57B887C887C88"
GC="6B69B1A794A894A86B"
[Host Extender Info]
&H00000001={3832D640-CF90-11CF-8E43-00A0C911005A};VBE;&H00000000
[Workspace]
ThisDocument=0, 0, 0, 0, C
Module1=26, 26, 388, 131, Z
```

It is interesting to note that we initially identified this actor only performing this manipulation on their malicious documents —avoiding any versioning complexities--without actually stomping the p-code to mismatch the VBA source. This seems like an odd decision and is possibly indicative of an actor assessing what "works" for their campaigns. The above malicious document is an example of them leveraging both methodologies, as highlighted by this screenshot from the awesome publicly available web service IRIS-H Digital Forensics:

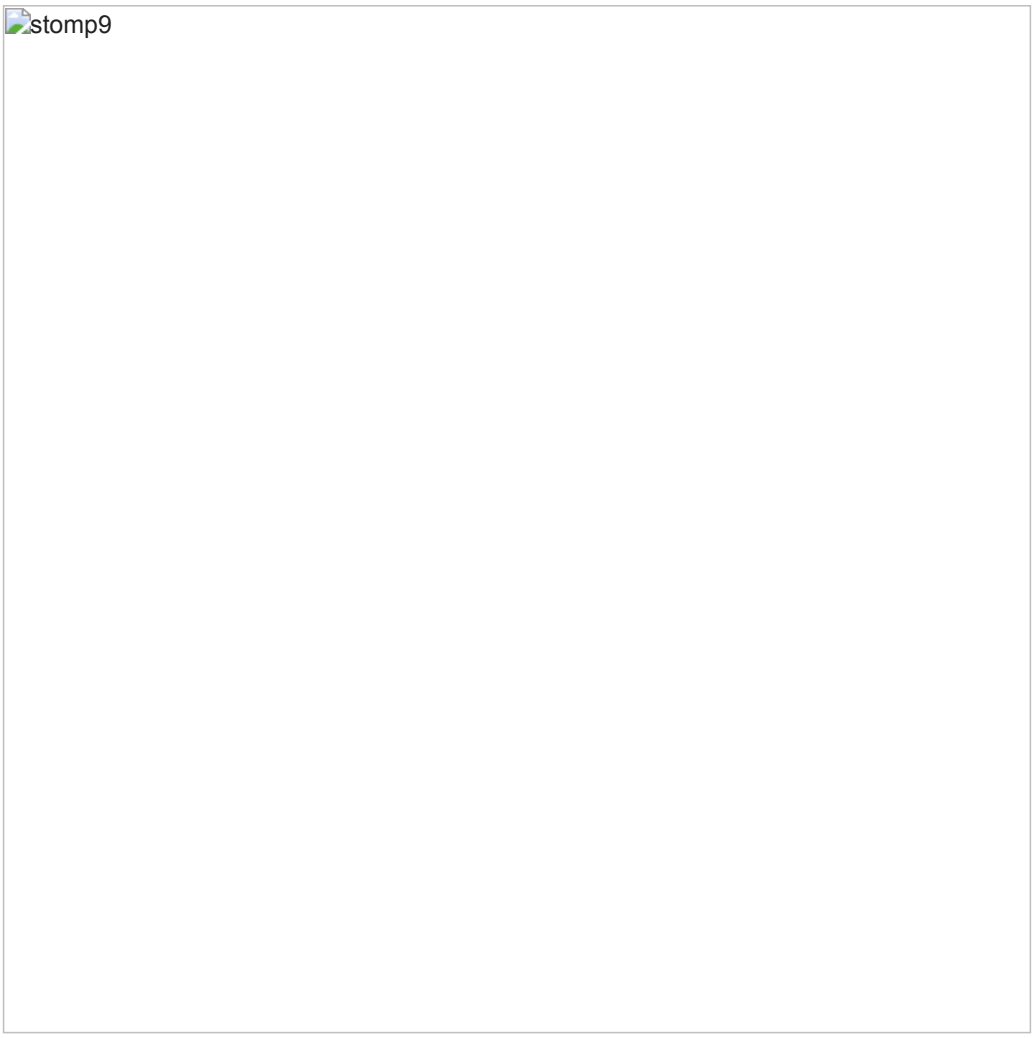


We can see that the documents VBA source is a blank Sub procedure definition. A quick glance at the p-code identifies both network- based indicators and host- based indicators we can use to determine what this sample would do when executed on the proper Office version. When we attempt to open the macro in the GUI editor, Office gets angry:

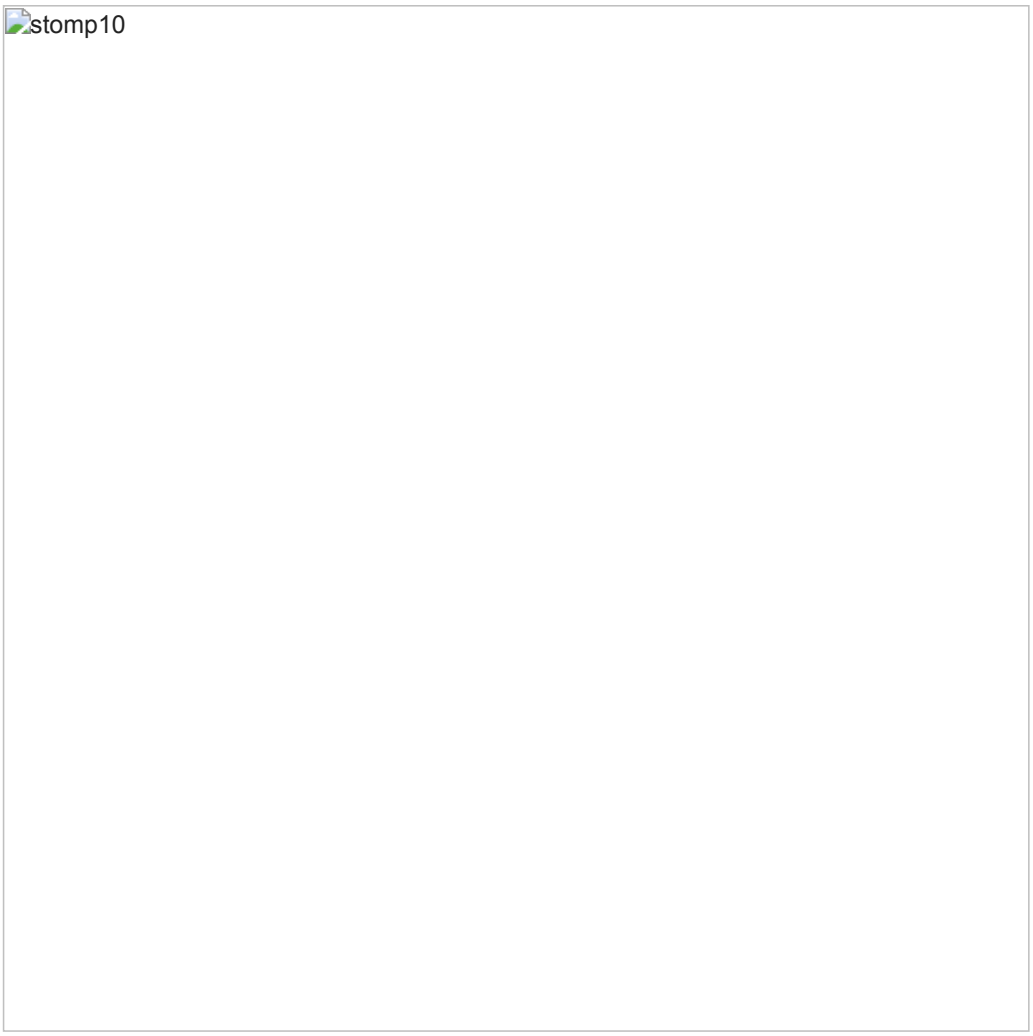


For analysts looking to identify this methodology holistically, we recommend the following considerations:

- The GUI hiding functionality results in an altered project stream wherein a module exists, but there is no module, class, or baseclass defined in the stream. This is a potential static detection.
- While the macro source is no longer present, there are still static strings present in Module1 in this sample which may indicate Windows APIs leveraged. This is a potential static detection.



Utilities like the previously mentioned oletools can do all of this detection for you. If you identify false negatives, false positives, or bugs, the open source project maintainers respond to them regularly like the superheroes that they are:

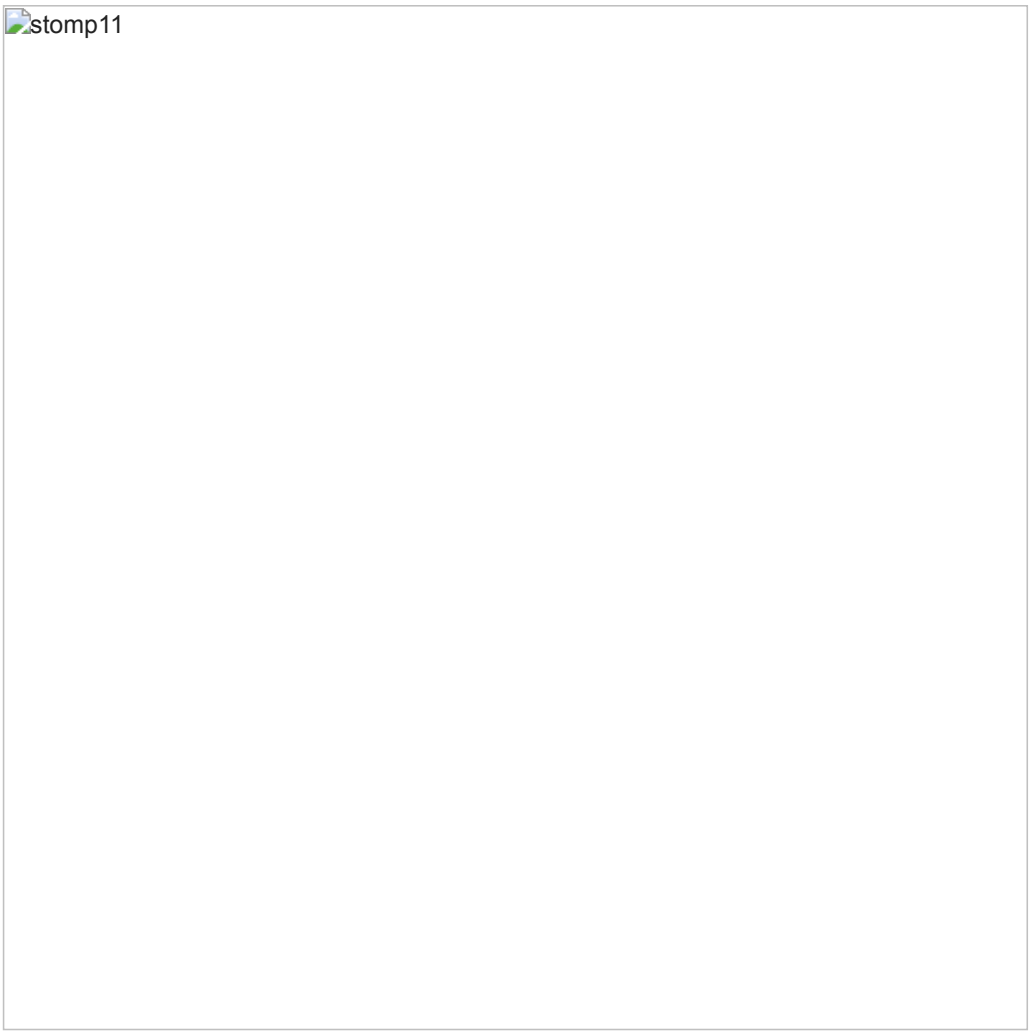


The above methodology creates questions regarding potential efficiency problems for scaling any sizable campaign using it. While tools like EvilClippy provide the means to create difficult to detect malicious documents that can potentially sneak past some dynamic and static detections, their payloads have the additional burden of needing to fingerprint targets to enable successful execution. While actors with sufficient resources and creativity can no doubt account for these requirements, it is relevant to note that detections for these methodologies will likely yield more targeted activity. In fact, tertiary review of samples employing these techniques identified unrelated activity delivering both Cobalt Strike BEACON and POSHC2 payloads.

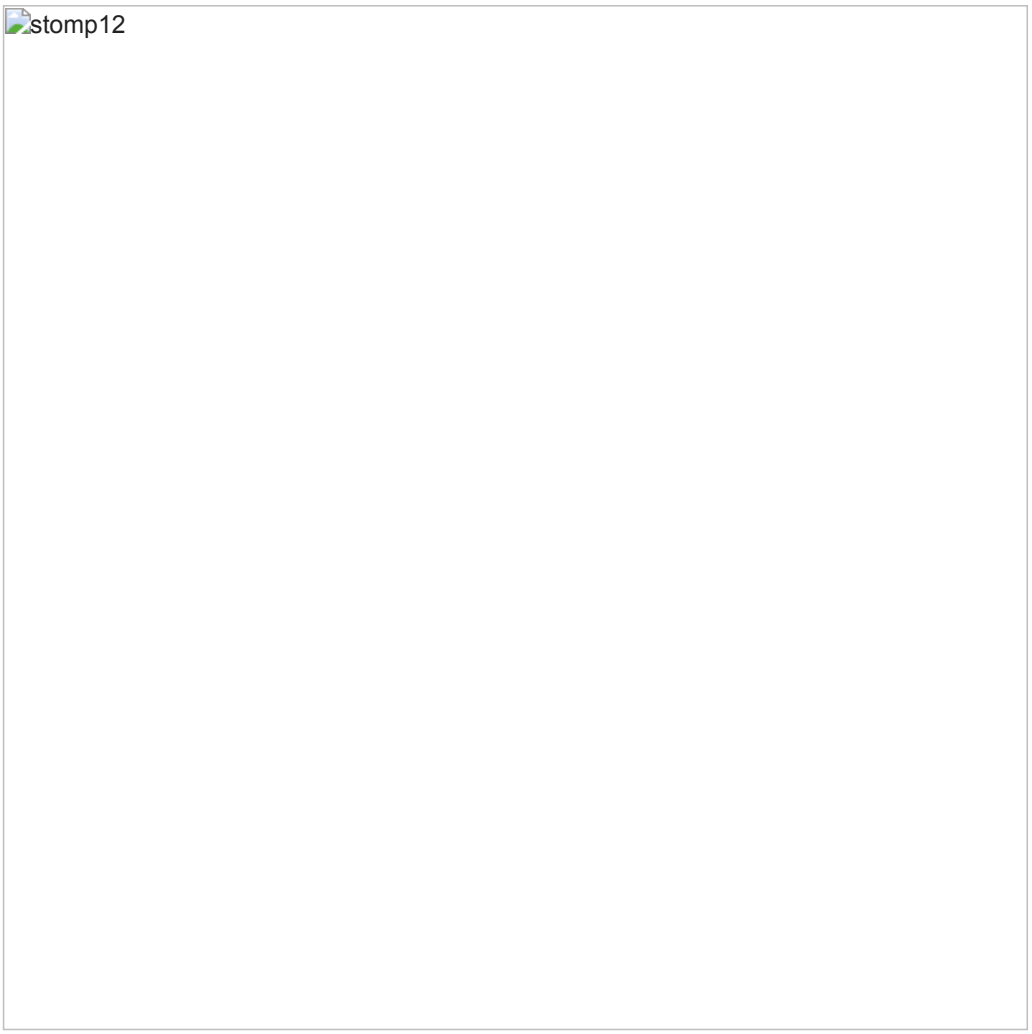
We recently expanded our internal FireEye threat behavior tree to accommodate these techniques. At the time of publication, the authors were unable to directly map the methods – PROJECT stream manipulation and VBA stomping – to existing techniques in the MITRE ATT&CK Matrix™ for Enterprise. However, our team submitted these as contributions to the ATT&CK knowledge base prior to publication and will make additional data available for ATT&CK Sightings.

Crossing The Bridge of Khazad-dûm: The MINEBRIDGE Infection Chain

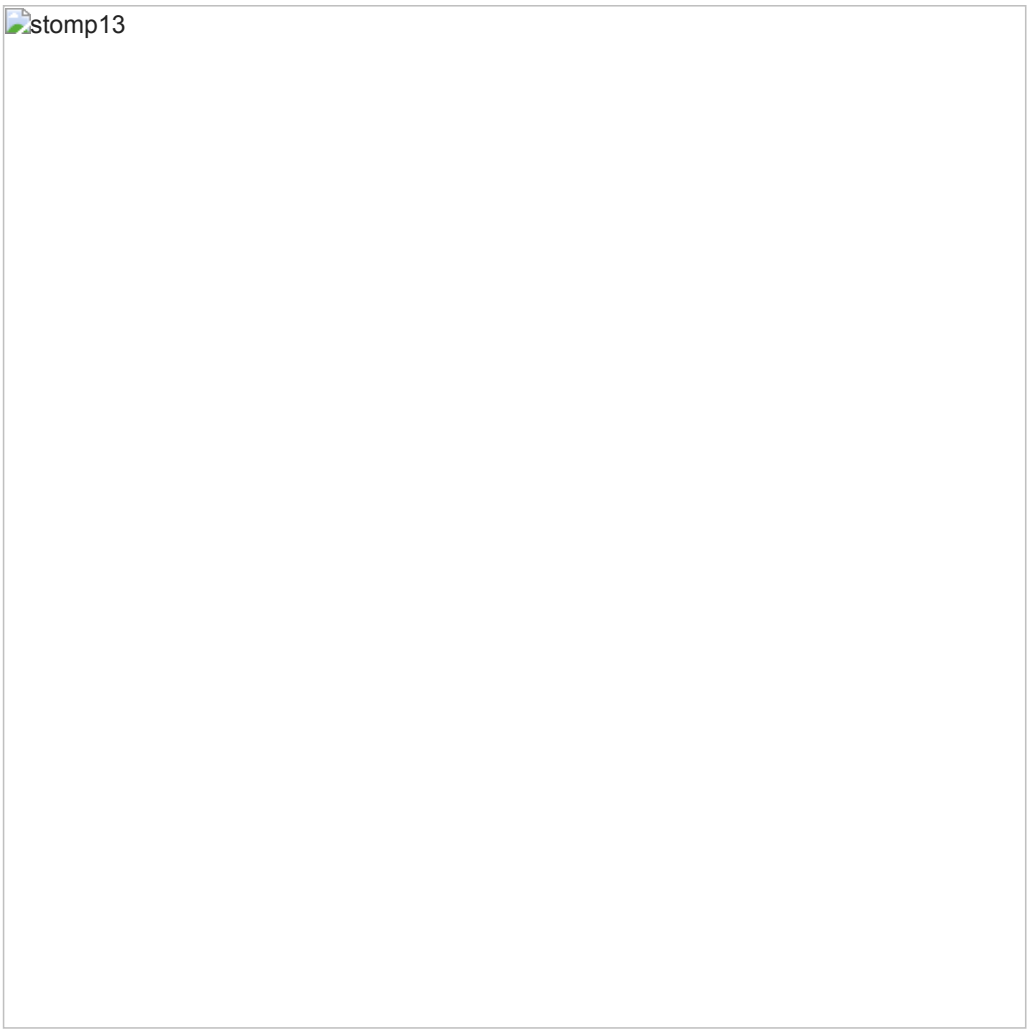
Successful detonation of the previously detailed malicious document results in creation of “uCWOncHvBb.dll” via a call to URLDownloadToFileA to the URL hxxps://marendoger[.]com/team/rumba.php. The returned MINEDOOR packed MINEBRIDGE sample is saved in the executing users AppData directory (Eg: C:\Users\username\AppData\Roaming\uCWOncHvBb.dll), and then subsequent execution of the DllRegisterServer export via invocation of “regsvr32.exe /s %AppData%\uCWOncHvBb.dll” occurs:



This will result in a ZIP file being retrieved from the URL hxxps://creatorz123[.]top/~files_tv/~all_files_m.bin using the Windows API URLDownloadToFileW. The ZIP file is written to %TEMP%, unzipped to the newly created directory %AppData%\Windows Media Player, and then deleted:



The ZIP file contains legitimate files required to execute a copy of TeamViewer, listed in the file creation area of the IOC section of this post. When a file named TeamViewer.exe is identified while unzipping, it is renamed to wpvnetwks.exe:



After completing these tasks, uCWOncHvBb.dll moves itself to %AppData%\Windows Media Player\msi.dll. The phishing macro then closes the handle to msi.dll, and calls CreateProcessA on wpvnetwks.exe, which results in the renamed TeamViewer instance side-loading the malicious msi.dll located alongside it. The malware ensures its persistence through reboot by creating a link file at %CISDL_STARTUP%\Windows WMI.lnk, which points to %AppData%\Windows Media Player\wpnetwks.exe, resulting in its launch at user logon.

The end result is a legitimate, though outdated (version 11, compiled on September 17, 2018, at 10:30:12 UTC), TeamViewer instance hijacked by a malicious sideloaded DLL (MINEBRIDGE).

MINEBRIDGE is a 32-bit C++ backdoor designed to be loaded by an older, unpatched instance of the legitimate remote desktop software TeamViewer by DLL load-order hijacking. The backdoor hooks Windows APIs to prevent the victim from seeing the TeamViewer application. By default, MINEBRIDGE conducts command and control (C2) communication via HTTPS POST requests to hard-coded C2 domains. The POST requests contain a GUID derived from the system's volume serial number, a TeamViewer unique id and password, username, computer name, operating system version, and beacon interval. MINEBRIDGE can also communicate with a C2 server by sending TeamViewer chat messages using a custom window procedure hook. Collectively, the two C2 methods support commands for downloading and executing payloads, downloading arbitrary files, self-deletion and updating, process listing, shutting down and rebooting the system, executing arbitrary shell commands, process elevation, turning on/off TeamViewer's microphone, and gathering system UAC information.

MINEBRIDGE's default method of communication is sending HTTPS POST requests over TCP port 443. This method of communication is always active; however, the beacon-interval time may be changed via a command. Before sending any C2 beacons, the sample waits to collect the TeamViewer generated unique id (<tv_id>) and password (<tv_pass>) via SetWindowsTextW hooks.

This specific sample continuously sends an HTTP POST request over TCP port 443 with the URI ~f83g7bfiunwjsd1/g4t3_indata.php to each host listed below until a response is received.

- 123faster[.]top
- conversia91[.]top
- fatoftheland[.]top
- creatorz123[.]top
- compilator333[.]top

The POST body contains the formatted string uuid=<guid>&id=<tv_id>&pass=<tv_pass>&username=<user_name>&pcname=<comp_name>&osver=<os_version>&timeout=<beacon_interval> where <guid> is a GUID derived from the system's volume serial number and formatted using the format string %06lX-%04lX-%04lX-%06lX. Additionally, the request uses the hard-coded HTTP User-Agent string "Mozilla/5.0 (iPhone; CPU iPhone OS 11_1_1 like Mac OS X) AppleWebKit/604.3.5 (KHTML, like Gecko) Version/11.0 Mobile/15B150 Safari/604.1"

After a response is received, it's processed for commands. A single response may contain multiple commands. For each command executed, the sample sends an HTTPS POST request over TCP port 443 indicating success or failure. The sample responds to the commands below.

Command	Description
drun	Download and execute an executable from a URL provided in the command. File saved to *%TEMP%*\<32_rand_chars>.exe.
rundll_command	Download a custom XOR-encoded and LZNT1 compressed DLL from a URL provided in the command and save to *%TEMP%*\<32_rand_chars>. Decode, decompress, and load the DLL in memory and call its entrypoint.
update_command	Move sample file to <sample_name>.old and download a new version of itself to <sample_name> where <sample_name> is the name of this sample (i.e., msi.dll). Relaunch the hosting TeamViewer application with command-line argument COM1_. Delete <sample_name>.old.
restart_command	Relaunch the hosting TeamViewer application with command-line argument COM1_.
terminate_command	Terminate the hosting TeamViewer application.
kill_command	Create and execute the self-deleting batch script tvdll.cmd to delete all unzipped files as well as the sample file. Terminate the hosting TeamViewer application.
poweroff_command	Shutdown the system.
reboot_command	Reboot the system.
setinterval_command	Update the C2 beacon-interval time.

After executing all commands in the response, the sample sleeps for the designated C2 beacon-interval time. It repeats the process outlined above to send the next C2 beacon. This behavior repeats indefinitely.

The self-deleting batch script tvdll.cmd contains the following content where <renamed_TeamVeiwer> is the renamed TeamViewer executable (i.e., wpvnetwks.exe) and <sample_name> is the name of this sample (i.e., msi.dll).

```
@echo off
ping 1.1.1.1 -n 1 -w 5000 > nul
goto nosleep1
:redel1
ping 1.1.1.1 -n 1 -w 750 > nul
:nosleep1
attrib -a -h -s -r %~d0%~p0TeamViewer_Resource_en.dll
del /f /q %~d0%~p0TeamViewer_Resource_en.dll
if exist  "%~d0%~p0TeamViewer_Resource_en.dll" goto redel1
goto nosleep2
:redel2
ping 1.1.1.1 -n 1 -w 750 > nul
:nosleep2
attrib -a -h -s -r %~d0%~p0TeamViewer_StaticRes.dll
del /f /q %~d0%~p0TeamViewer_StaticRes.dll
if exist  "%~d0%~p0TeamViewer_StaticRes.dll" goto redel2
goto nosleep3
:redel3
ping 1.1.1.1 -n 1 -w 750 > nul
:nosleep3
attrib -a -h -s -r %~d0%~p0TeamViewer_Desktop.exe
del /f /q %~d0%~p0TeamViewer_Desktop.exe
if exist  "%~d0%~p0TeamViewer_Desktop.exe" goto redel3
goto nosleep4
:redel4
ping 1.1.1.1 -n 1 -w 750 > nul
:nosleep4
attrib -a -h -s -r %~d0%~p0TeamViewer.ini
del /f /q %~d0%~p0TeamViewer.ini
if exist  "%~d0%~p0TeamViewer.ini" goto redel4
goto nosleep5
:redel5
ping 1.1.1.1 -n 1 -w 750 > nul
:nosleep5
attrib -a -h -s -r %~d0%~p0<sample_name>
del /f /q %~d0%~p0<sample_name>
if exist  "%~d0%~p0<sample_name>" goto redel5
goto nosleep6
:redel6
ping 1.1.1.1 -n 1 -w 750 > nul
:nosleep6
attrib -a -h -s -r %~d0%~p0<renamed_TeamVeiwer>
del /f /q %~d0%~p0<renamed_TeamVeiwer>
if exist  "%~d0%~p0<renamed_TeamViewer>" goto redel6
attrib -a -h -s -r %0
del /f /q %0
```

Possible Connection to Another Intrusion Set

The identified MINEBRIDGE samples have been packed within a loader we call MINEDOOR. Since Fall 2019, we've observed a group publicly tracked as TA505 conducting phishing campaigns that use MINEDOOR to deliver the FRIENDSPEAK backdoor. The combination of MINEDOOR and FRIENDSPEAK has also been publicly discussed using the name Get2.

The limited overlap in tactics, techniques, and procedures (TTPs) between campaigns delivering MINEBRIDGE and those delivering FRIENDSPEAK may suggest that MINEDOOR is not exclusive to TA505. Recent campaigns delivering FRIENDSPEAK have appeared to use spoofed sender addresses, Excel spreadsheets with embedded payloads, and campaign-specific domains that masquerade as common technology services. Meanwhile, the campaigns delivering MINEBRIDGE have used actor-controlled email addresses, malicious Word documents that download payloads from a remote server, and domains with a variety of themes sometimes registered weeks in advance of the campaign. The campaigns delivering MINEBRIDGE also appear to be significantly smaller in both volume and scope than the campaigns

delivering FRIENDSPEAK. Finally, we observed campaigns delivering MINEBRIDGE on Eastern Orthodox Christmas when Russian-speaking actors are commonly inactive; we did not observe campaigns delivering FRIENDSPEAK during the week surrounding the holiday and language resources in the malware may suggest TA505 actors speak Russian.

It is plausible that these campaigns represent a subset of TA505 activity. For example, they may be operations conducted on behalf of a specific client or by a specific member of the broader threat group. Both sets of campaigns used domains that were registered with Eranet and had the registrant location “JL, US” or “Fujian, CN,” however this overlap is less notable because we suspect that TA505 has used domains registered by a service that reuses registrant information.

Post-compromise activity would likely reveal if these campaigns were conducted by TA505 or a second threat group, however, FireEye has not yet observed any instances in which a host has been successfully compromised by MINEBRIDGE. As such, FireEye currently clusters this activity separately from what the public tracks as TA505.

Acknowledgments

FireEye would like to thank all the dedicated authors of open source tooling and research referenced in this blog post. Further, FireEye would like to thank TeamViewer for their collaboration with us on this matter. The insecure DLL loading highlighted in this blog post was resolved in TeamViewer 11.0.214397, released on October 22, 2019, prior to the TeamViewer team receiving any information from FireEye. Additionally, TeamViewer is working to add further mitigations for the malware’s functionality. FireEye will update this post with further data from TeamViewer when this becomes available.

Indicators of Compromise (IOCs)

Suspicious Behaviors

- Process lineage: Microsoft Word launching TeamViewer
- Directory Creation: %APPDATA%\Windows Media Player
- File Creation:
  - %APPDATA%\Windows Media Player\msi.dll
  - %APPDATA%\Windows Media Player\msi.dll.old
  - %APPDATA%\Windows Media Player\tvdll.cmd
  - %APPDATA%\Windows Media Player\wpvnetwks.exe
  - %APPDATA%\Windows Media Player\TeamViewer_Resource_en.dll
  - %APPDATA%\Windows Media Player\TeamViewer_StaticRes.dll
  - %APPDATA%\Windows Media Player\TeamViewer_Desktop.exe
  - %APPDATA%\Windows Media Player\TeamViewer.ini
  - %CSIDL_STARTUP%\Windows WMI.lnk
  - %CSIDL_PROFILE%\<dll_name>.xpdf
  - %TEMP%\<32 random characters>
  - %TEMP%\<32 random characters>.exe
  - %TEMP%\~8426bcrtv7bdf.bin
- Network Activity:
  - HTTPS Post requests to C2 URLs
  - User-Agent String: "Mozilla/5.0 (iPhone; CPU iPhone OS 11_1_1 like Mac OS X) AppleWebKit/604.3.5 (KHTML, like Gecko) Version/11.0 Mobile/15B150 Safari/604.1"

C2 Domains

- 123faster[.]top
- conversia91[.]top
- fatoftheland[.]top
- creatorz123[.]top
- compilator333[.]top

Download Domains

- neurogon[.]com
- tiparcano[.]com

- seigortan[.]com
- marendoger[.]com
- badiconreg[.]com

Sender Domains

- pt-cpaaccountant[.]com
- rogervecpa[.]com
- agent4career[.]com
- bestrecruitments[.]com

Phishing Documents

MD5	SHA256
01067c8e41dae72ce39b28d85bf923ee	80e48391ed32e6c1ca13079d900d3afad62e05c08bd6e929dffdd2e3b9f69299
1601137b84d9bebf21dcfb9ad1eaa69d	3f121c714f18dfb59074cbb665ff9e7f36b2b372cfe6d58a2a8fb1a34dd71952
1c883a997cbf2a656869f6e69ffbd027	de7c7a962e78ceeee0d8359197daeb2c3ca5484dc7cf0d8663fb32003068c655
2ed49bd499c9962e115a66665a6944f6	b8f64a83ad770add6919d243222c62471600e64789264d116c560b7c574669ec
3b948368fe1a296f5ed18b11194ce51c	999d4f434bbc5d355656cc2a05982d61d6770a4c3c837dd8ec6aff8437ae405a
4148281424ff3e85b215cd867746b20c	9812123d2367b952e68fa09bd3d1b3b3db81f0d3e2b3c03a53c21f12f1f4c889
54f22fbc84f4d060fcbf23534a02e5f6	7b20e7e4e0b1c0e41de72c75b1866866a8f61df5a8af0ebf6e8dbd8f4e7bdc57
5a3d8348f04345f6687552e6b7469ac1	77a33d9a4610c4b794a61c79c93e2be87886d27402968310d93988dfd32a2ccf
607d28ae6cf2adb87fcb7eac9f9e09ab	f3917832c68ed3f877df4cd01635b1c14a9c7e217c93150bebf9302223f52065
9ba3275ac0e65b9cd4d5afa0adf401b4	18698c5a6ff96d21e7ca634a608f01a414ef6fbbd7c1b3bf0f2085c85374516e
9becd2fd73aa4b36ad9cd0c95297d40b	30025da34f6f311efe6b7b2c3fe334f934f3f6e6024e4d95e8c808c18eb6de03
9cce3c9516f0f15ce18f37d707931775	bf0adb30ca230eee6401861e1669b9cfeaa64122cc29c5294c2198f2d82f760e
9faf9e0c5945876c8bad3c121c91ea15	88c4019e66564ad8c15b189b903276910f9d828d5e180cac30f1f341647278fc
a37e6eeb06729b6108649f21064b16ef	e895dc605c6dcaf2c3173b5ec1a74a24390c4c274571d6e17b55955c9bd48799
ab8dc4ba75aad317abb8ee38c8928db0	212793a915bdd75bede8a744cd99123e2a5ac70825d7b2e1fc27104276a3aafd
b8817253288b395cb33ffe36e0072dc9	ba013420bd2306ecb9be8901db905b4696d93b9674bd7b10b4d0ef6f52fbd069
cb5e5d29f844eb22fecaa45763750c27	4ff9bfde5b5d3614e6aa753cacc68d26c12601b88e61e03e4727ee6d9fe3cdc2
cffda37453e1a1389840ed6ebaef1b0d	c9f6ba5368760bf384399c9fd6b4f33185e7d0b6ea258909d7516f41a0821056

dc0e1e4ec757a777a4d4cc92a8d9ef33	ac7e622e0d1d518f1b002d514c348a60f7a7e7885192e28626808a7b9228eab6
e5c7e82670372e3cf8e8cab2c1e6bc17	eba3c07155c47a47ee4d9b5201f47a9473255f4d7a6590b5c4e7b6e9fc533c08
f93062f6271f20649e61a09c501c6c92	3f4f546fba4f1e2ee4b32193abcaaa207efe8a767580ab92e546d75a7e978a0b

MINEBRIDGE/MINEDOOR Samples

MD5	**SHA256**
05432fc4145d56030f6dd6259020d16c	182ccc7f2d703ad732ffee0e1d9ae4ae5cf6b8817cc33fd44f203d31868b1e97
0be9911c5be7e6dfeaeca0a7277d432b	65ead629a55e953b31668aac3bd373e229c45eb1871d8466f278f39ebcd5d26b
0dd556bf03ecb42bf87d5ea7ce8efafe	48f6810e50d08c2631f63aae307a7724dba830430f5edd4b90b4b6a5b3c3ca85
15edac65d5b5ed6c27a8ac983d5b97f6	03ff2b3067aa73ecd8830b6b0ea4f7cfa1c7476452b26227fb433265e7206525
1e9c836f997ddcbd13de35a0264cf9f1	23da418912119a1358c9a1a4671ba60c396fff4c4de225fe6a225330147549a7
21aa1066f102324ccc4697193be83741	86d839e1d741445f194965eee60d18bd292bec73e4889089e6caf9877581db12
22b7ddf4983d6e6d84a4978f96bc2a82	fc39cb08cae90c661e00718e2a0051b5de3dcb7cddde919b9ffd2d79bf923d1f
2333fbadeea558e57ac15e51d55b041c	57671d5154e707da0ee6139485f45a50fa9221852ebb65781d45a2660da7d0cb
2b9961f31e0015cbcb276d43b05e4434	e41b89869c2b510c88acd1ed9fd4a6dfe89222a81c6c1241a69af3b7f812f712
2c3cb2132951b63036124dec06fd84a8	b6dbb902125e7bf6f6701b654cbff4abaf2e853441cf34045ac19eff5ed8ce84
4de9d6073a63a26180a5d8dcaffb9e81	7b1d4774176976ffcb2075889557f91a43c05fb13f3bc262bbaec4d7a0a827e6
505ff4b9ef2b619305d7973869cd1d2b	abb05ba50f45742025dd4ebff2310325783da00fb7bc885783e60a88c5157268
52d6654fe3ac78661689237a149a710b	d6a0e62fe53116c9b5bccd2a584381e2ca86e35490d809ce1900603d5e6b53eb
53e044cd7cea2a6239d8411b8befb4b7	6e76d648d446e6a70acdd491f04c52d17f9f0e1ef34890c6628c4f48725b47c8
5624c985228288c73317f2fa1be66f32	99559a5f06b0279ed893d2799b735dae450a620f6cea2ea58426d8b67d598add
598940779363d9f4203fbfe158d6829b	1358b0ccae9dbb493228dc94eb5722c8d34c12227a438766be83df8c1c92a621
60bdea2c493c812428a8db21b29dd402	383c86deed8797e0915acf3e0c1b6a4142c2c5ecb5d482517ed2ade4df6f36fd
681a77eba0734c0a17b02a81564ae73f	0aaa66dc983179bffdb181079f3b786b6cd587c38c67ba68b560db0bd873278a

6b7d9268c7000c651473f33d088a16bd	6e39ffecab4ca0bd7835a2e773ebfc3f6d909a0a680f898e55f85ed00728666d
6d6f50f7bba4ae0225e9754e9053edc0	ddf33eff293ffc268dfd0a33dddef97aefe9e010ec869dc22c221d197eb85740
6de77c1b4e8abaaf304b43162252f022	8f50ddc1519e587597882a6bd0667653c36a8064b56ee5ff77665db2faf24710
7004fadfa572d77e24b33d2458f023d1	cccd6b46f950caec5effdd07af339be78691974fec5f25d923932b35edb95c4a
71988460fd87b6bff8e8fc0f442c934b	8167d41ad30f5d451791878815e479965b2f5213231f26819ecaf4fcc774ab12
722981703148fa78d41abbae8857f7a2	a3070ee10dd5bcd65a45b72848c926db2602e5297641452edff66e7133cdce9c
818f7af373d1ec865d6c1b7f59dc89e5	cbe4b73c0c95c207ccde9d9bd80f541cf90cad18ba5abc3fe66a811ead1601c2
832052b0f806f44b92f6ef150573af81	e162a70a6e27fe23379d3a17a3a727d85a94b79416d81ec3b4ea80d329e96830
836125ae2bed57be93a93d18e0c600e8	0fbde653bef4642626f2996a41a15a635eb52cd31eacce133d28301b902d67df
86d60bce47c9bb6017e3da26cab50dcf	6c134908ad74dfa1468a1166e7d9244695f1ffeff68bfd4eec4b35820b542b8a
8919458aec3dcc90563579a76835fc54	aad0537924bacddd0d5872f934723e765dbb182f2804c6f594f9b051937495ec
8d7e220af48fceee515eb5e56579a709	3eefa7072344e044c0a6abb0030f3f26065bf6a86bb50ea38473dd7ac73904fb
91b8ec04d8b96b90ea406c7b98cc0ad6	0520e68a4b73c3b41e566cf07be54e1f1cb59c59c303fe3390e0687f9af1a58a
959eb0696c199cbf60ec8f12fcf0ea3c	ccb5f8734befd6ab218513e16a57679a8fb43b2732e19233ee920d379045e318
95ec5e8d87111f7f6b2585992e460b52	3f8e38ccf71f122b65fdc679db13e3de3bb4b4fc04b8ab6f955d02e0bca10fae
9606cf0f12d6a00716984b5b4fa49d7d	f4f062fd7b98365ed6db993b1da586dd43e5cdcc2f00a257086734daf88c9abb
9f7fed305c6638d0854de0f4563abd62	6c5f72ddf0262838a921107520cdc12ba8e48dbafab4a66732a350095dd48e9f
a11c0b9f3e7fedfe52b1fc0fc2d4f6d1	d35ac29ea6e064b13d56f6a534022f253cf76b98e10a7ea1cbfa086eefd64f4b
a47915a2684063003f09770ba92ccef2	7b16ce0d2443b2799e36e18f60fe0603df4383b1a392b0549c3f28159b1ca4d4
a917b2ec0ac08b5cde3678487971232a	8578bff803098bf5ca0d752d0a81f07659688a32cbfc946728e5ab0403f5c4ba
ad06205879edab65ed99ed7ff796bd09	d560f8717f4117d011f40c8880081d02d1455a41c93792e1600799d3e5ee9421
ad910001cb57e84148ef014abc61fa73	c9a6f7b0603779690c1d189850403f86608a3c5e1cd91e76fd31c4f119ae256b
b1ce55fca928cf66eaa9407246399d2c	c6214ec7909ce61d6ec3f46f5a7ec595d8cc8db48965c5baee8a346632cbe16d

b9249e9f1a92e6b3359c35a8f2a1e804	0695e5e49a297c980b96f76bf10e5540de188d6a6a162e38f475418d72a50032
bd6880fb97faceecf193a745655d4301	23840c587e4e9588b3d0795d4d76a4f3d4d5b2e665ce42dde0abcd1e0a2ba254
be2597a842a7603d7eb990a2135dab5e	6288d3de1f1aa05fa0a5f0c8eb9880d077f034fc79fc20f87cbfcc522aa803cb
cf5470bfe947739e0b4527d8adb8486a	6357fdb8f62948d489080b61caf135e6aaba32dcdb7dc49b0efafef178b3b54f
d593b7847ec5d18a7dba6c7b98d9aebf	5df3a6afb1a56fa076c6db716d5a050455158941ec962546a8799fc80ccfa573
d7ee4ffce21325dfe013b6764d0f8986	92e94482dee75261c8ebdcbb7ace382a097cca11bcdc675bbe2d7b3f67525f84
de4d7796006359d60c97a6e4977e4936	ee8ba1c5329d928d542bfa06eec2c0a3e3b97dcc20382ddbc27bc420ceaeb677
e0069cd3b5548f9fd8811adf4b24bf2e	6046d6aed3f4ee2564d6be540d46bcdc0bebce11a1ced4b9ddbfa1a41084411c
e1ea93fa74d160c67a9ff748e5254fe0	92c10ef23209e09abb17e41d67301f0e3f7d9e7ddfc7c1a66140c4986d72bee7
ea15d7944c29f944814be14b25c2c2b1	5898b41ca4f4777ad04d687f93548129ccb626d2f5e6e100b0a037c3d40a7444
f22a4abd5217fa01b56d064248ce0cc5	858b4070f8b83aa43fd6a5189a8ed226ce767a64972db893e36550a25b20be94
f3cb175e725af7f94533ecc3ff62fa12	5a5385df469459cd56f6eecbf4b41b8c75aa17220c773501eaec22731f3a41bb
f6533e09a334b9f28136711ea8e9afca	9136c36ccd0be71725e8720a6cfdbdd38d7eea3998228c69ed4b52e78ba979c4
f7daaea04b7fe4251b6b8dabb832ee3a	6abd90d718113482a5bcd36e35b4ea32c469f94fc2cfb9c1c98214efbf64c352
fb1555210d04286c7bcb73ca57e8e430	36da56815dc0c274fc8aacdfffbc4d5e500025ccd1147cad513d59b69ab9557d