

# Salfram: Robbing the place without removing your name tag

By Holger Unterbrink

Published: 2020-09-03 · Archived: 2026-04-05 23:36:18 UTC



Thursday, September 3, 2020 11:05

## Threat summary

- Cisco Talos recently uncovered a series of email campaigns utilizing links to malicious documents hosted on legitimate file-sharing platforms to spread malware.
- The campaigns distributed various malware payloads including [Gozi ISFB](#), ZLoader, [SmokeLoader](#) and AveMaria, among others.
- Ongoing campaigns are distributing various malware families using the same crypter. While effective, this crypting mechanism contains an easy-to-detect flaw: The presence of a specific string value "Salfram" makes it easy to track over time.
- Obfuscated binaries are completely different, from both a binary and execution flow graph perspective.
- The techniques used by this crypter can confuse weak API behavior-based systems and static analysis tools.
- This crypter appears to be undergoing active development and improvement over time.

## Executive summary

Over the past several months, Cisco Talos has seen attackers carrying out ongoing email-based malware distribution campaigns to distribute various malware payloads. These email campaigns feature several notable characteristics that appear designed to evade detection and maximize the effectiveness of these campaigns. The use of web-based contact forms, legitimate hosting platforms, and a specific crypter make analysis and detection

more difficult. All of the malware samples associated with these campaigns feature a modified DOS header containing the string "Salfram," making them extremely easy to track over time. The crypter used in these campaigns is undergoing active development and improvements to obfuscate the contents of malware payloads. Additionally, the crypter uses several effective techniques to make the detection and analysis of the final malware payload more difficult. It obfuscates the original payload binaries in a way that results in payloads that appear completely different from each other after being packed using the same crypter. It even takes a great amount of effort to compare the packed binaries in a disassembler and determine if the same packer was used.

## Distribution campaigns

The email distribution campaigns associated with these malware payloads feature characteristics designed to evade detection and make analysis and tracking more difficult. In most cases, the threat actor was initiating email-based communications with potential victims using the contact forms present on the victim organization's website. This results in email communications that appear to originate from legitimate sources which may allow the adversaries to evade some email security mechanisms.

In the messages, the attacker purports to be the copyright owner of images that the attacker claims they found hosted on the target organization's website without their permission. The email contents are somewhat customized, listing the target organization's domain as the website containing the infringing content and prompting them to access a hyperlink present in the email. Figure 1 is an example of one of the emails associated with one of these campaigns.

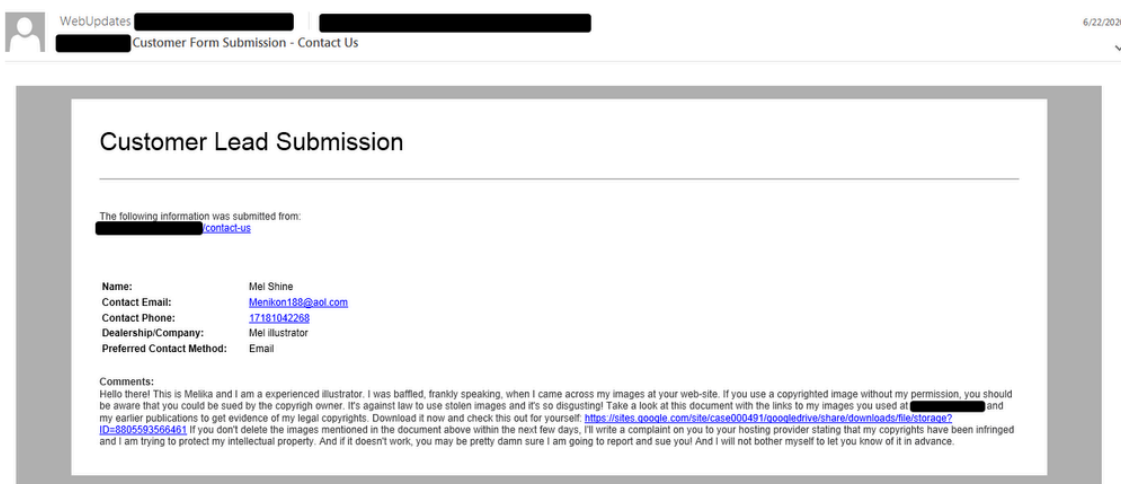


Figure 1

Figure 2 is another example of these emails, demonstrating how the contents of individual messages may change based on the contact form and website configuration for the organization being targeted.

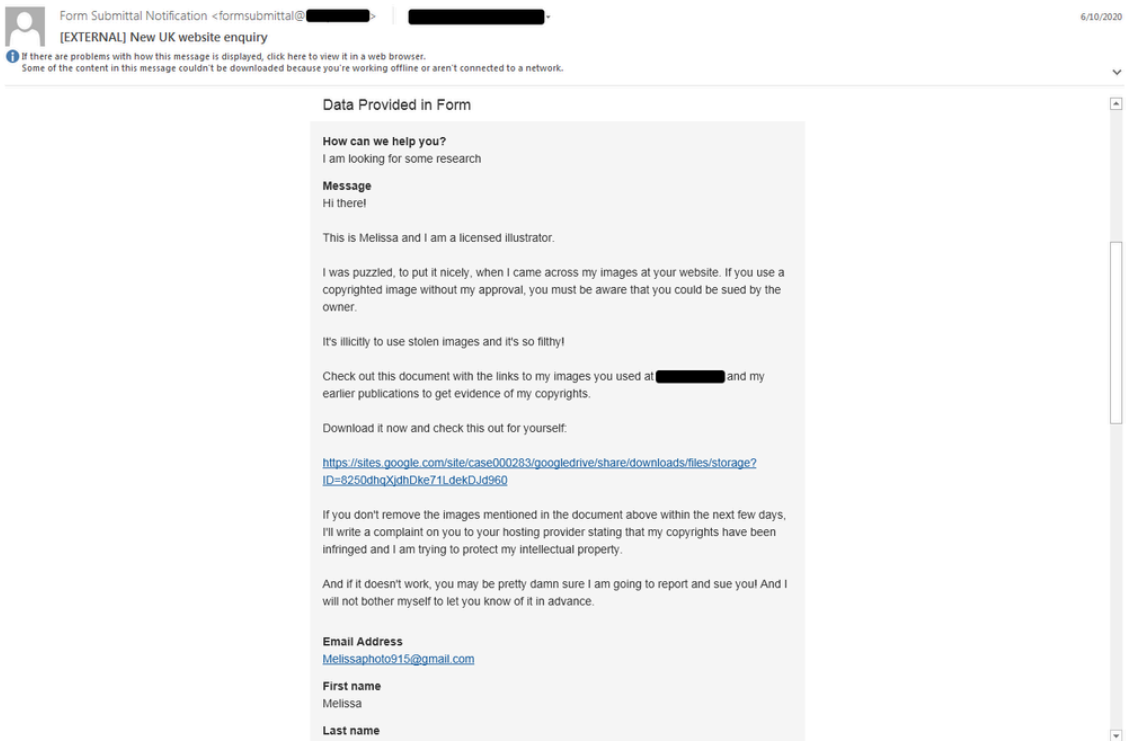


Figure 2

In analyzing malicious contact form submissions associated with these campaigns, we obtained information about the system used to submit the form contents. The contact form submissions were performed via proxy servers using an automated mechanism, such as scripting, to generate and submit the data.

The majority of the emails contained links to a malicious file hosted on Google Drive. The directory structure and parameter values rotate frequently across various emails. Figure 3 lists some examples of these URLs.

```
https://sites.google.com/site/case53703/googledrive/share/downloads/file/storage?ID=1076540191359
https://sites.google.com/site/case4909/googledrive/share/downloads/file/storage?ID=9374300554579
https://sites.google.com/site/case4909/googledrive/share/downloads/file/storage?ID=2612219293595
https://sites.google.com/site/case13703/googledrive/share/downloads/file/storage?ID=7731619769526
https://sites.google.com/site/case4909/googledrive/share/downloads/file/storage?ID=0873303935860
https://sites.google.com/site/case4909/googledrive/share/downloads/file/storage?ID=6316245887196
https://sites.google.com/site/case4909/googledrive/share/downloads/file/storage?ID=2746259314740
https://sites.google.com/site/case53703/googledrive/share/downloads/file/storage?ID=0535102798262
https://sites.google.com/site/case53703/googledrive/share/downloads/file/storage?ID=0535102798262
https://sites.google.com/site/case53703/googledrive/share/downloads/file/storage?ID=1624108228463
https://sites.google.com/site/case53703/googledrive/share/downloads/file/storage?ID=1684689091810
https://sites.google.com/site/case53703/googledrive/share/downloads/file/storage?ID=8023128209845
https://sites.google.com/site/case13703/googledrive/share/downloads/file/storage?ID=7953881985669
https://sites.google.com/site/case53703/googledrive/share/downloads/file/storage?ID=5075792523872
https://sites.google.com/site/case53703/googledrive/share/downloads/file/storage?ID=5223727819914
https://sites.google.com/site/case4909/googledrive/share/downloads/file/storage?ID=5659937714930
```

Figure 3

Again, the use of a legitimate web platform for hosting the malicious content may provide another way for the attacker to evade various protections that may be deployed in environments that they are targeting. When the URL is accessed, the victim is delivered a malicious file, often a Microsoft Office document similar to the one in Figure 4.

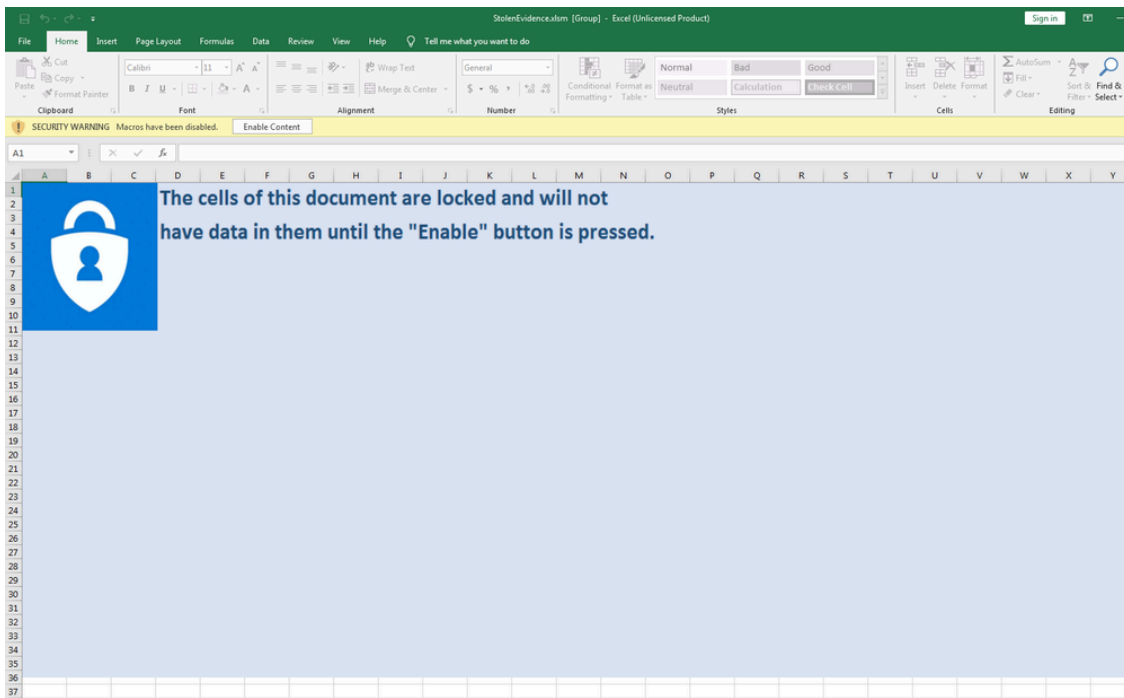


Figure 4

These documents contain malicious macros that initiate the infection process and are responsible for retrieving a malicious payload and executing it to infect the system. The payloads delivered have varied over time. Over the past several months, we have observed campaigns being used to deliver various information stealers, banking trojans and other malware loaders including but not limited to [Gozi ISFB](#), ZLoader, [SmokeLoader](#), Oski, AveMaria and Cobalt Strike payloads, among others.

We also discovered infection chains that deliver multiple payloads to a single victim as part of a multi-stage infection process. While the payload delivery features several distinct malware families, in all of the campaigns observed, the initial malware payloads used the same crypter, which obfuscates the malicious contents present in the binary executable and make analysis more difficult. As shown in Figure 5 these samples can be quickly observed as the packed samples feature a specific modification to the DOS stub:

```
00000000: 4d5a 9000 0300 0000 0400 0000 ffff 0000  MZ.....
00000010: b800 0000 0000 0000 4000 0000 0000 0000  .....@.....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000030: 0000 0000 0000 0000 0000 0000 8000 0000  .....
00000040: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468  .....!..L.!Th
00000050: 6973 2053 616c 6672 616d 2063 616e 6e6f  is Salfram canno
00000060: 7420 6265 2072 756e 2069 6e20 444f 5320  t be run in DOS
00000070: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000  mode...$.
00000080: 5045 0000 4c01 0300 6b67 c45e 0000 0000  PE..L...kg.^...
```

Figure 5

The next section describes how these techniques have been implemented and how to analyze samples that may have been obfuscated using this crypter.

## Technical overview

The obfuscation techniques used by this packer are not trivial, so it is surprising that all of the binaries observed contained the easily-trackable string "Salfram" inside the DOS stub. Further investigation is required to determine if the packer is automatically modifying contents of the DOS stub or if this is being done by the actor behind the campaigns.

The samples we analyzed have compiler or debug timestamps ranging from April 2019 to August 2020, but the majority of samples are from January 2020 onward. The campaigns before are either quite small or just tests. The binary structure of many of the samples is different, with differences ranging from minor changes all the way to code execution flows that are completely different. Nevertheless, they share the same or at least a very similar logic. It is hard to say if these changes are due to the use of different obfuscation features available within the crypter, or the use of different versions of the crypter. Comparing the samples we analyzed, we believe it is a bit of both — the crypter is frequently updated and actively maintained by the software author.

The following section describes a subset of the typical differences and similarities which we have observed when analyzing samples leveraging this obfuscator.

The majority of the samples begin with a common initialization routine (security\_init\_cookie) and then jump to the main unpacking function. This function may vary in different samples and even the start up looks different in the newer samples. Figures 6 - 8 show the startup and unpacking functions in three different samples.

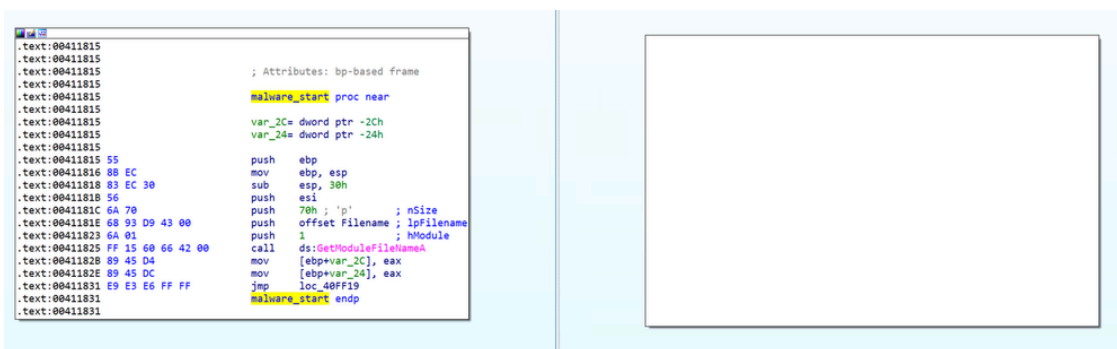


Figure 6: Sample 4bfd547775ba5892e66d2ff6a0c1de4365ab11b3\_exe.bin (946d4d332a06b9af10da38beb3e8195054840b59a870a2f9027e6471f4869dc6)

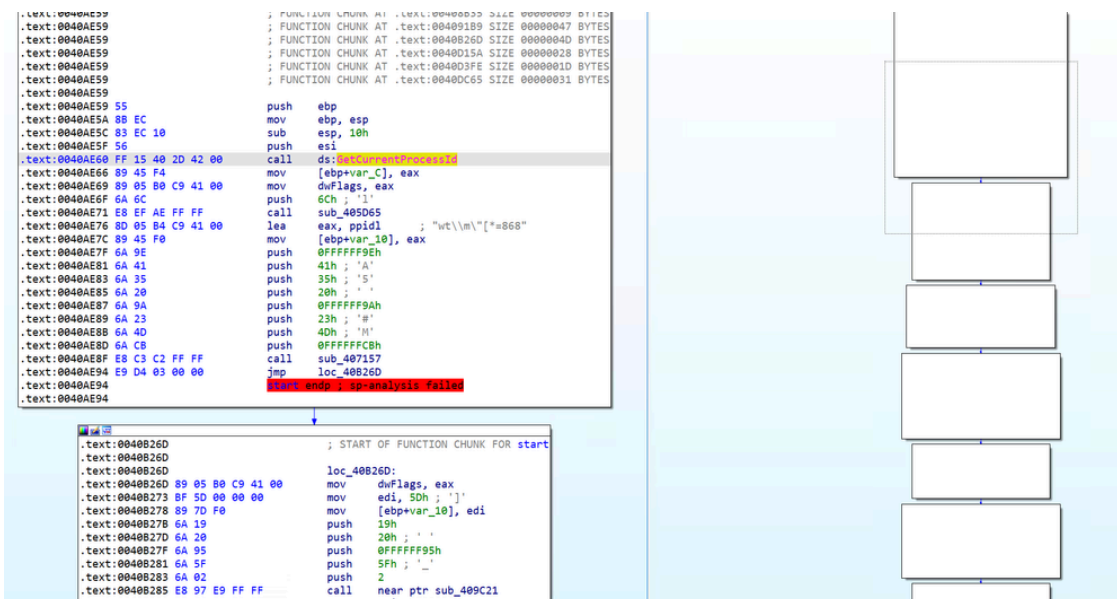


Figure 7: Sample 9679f2690d31cee38e57f080656b8618ca474c65\_exe.bin  
(9b28aa737bbfec90341c6a42e2d44f3308659e4fb9dd42d98a0b46cde7aaed63)

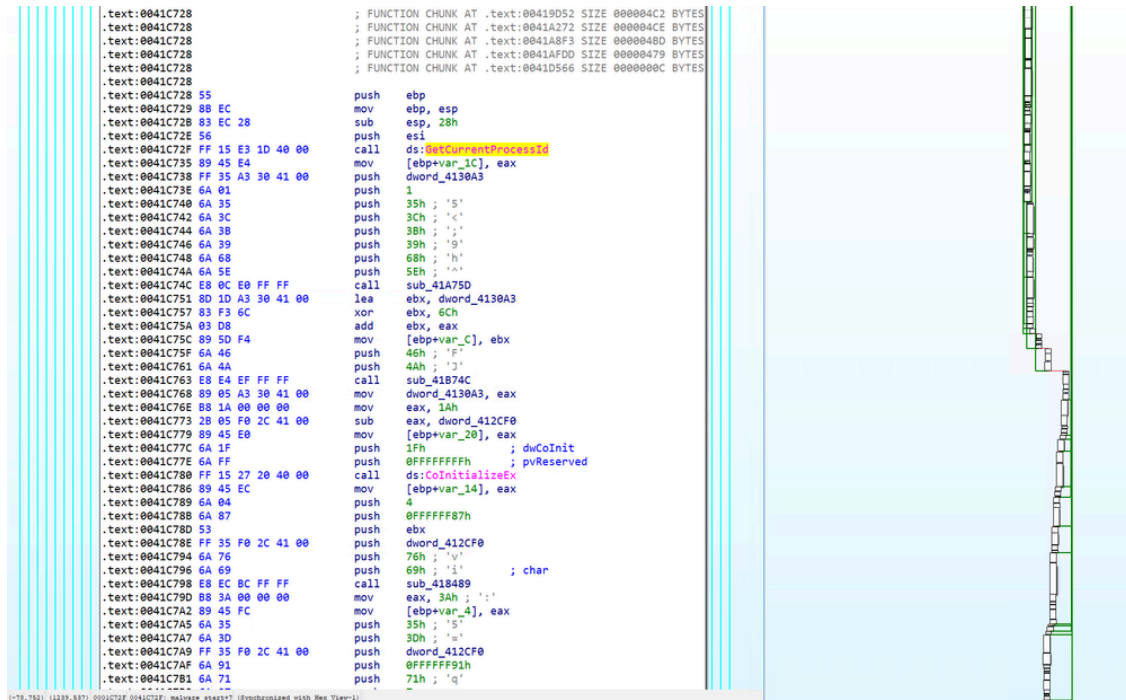


Figure 8: Sample 64800a2f85e12090eea5109e05d969a08ae9850a\_exe.bin  
(0c09ea2d5722a2484804bf8c4f051d66d8a95d23c19e5d1277a55cafd80b67be)

What they all have in common is that they push many unpacking-related values to the stack and insert a large number of fake API calls in between. Figure 6 shows the nonsense arguments handed over to GetModuleFileNameA. Figure 9 shows another example of this:

```

.text:00416332      loc_416332:
.text:00416332  6A 1F      push    1Fh
.text:00416334  6A 99      push    0FFFFFF99h
.text:00416336  50        push    eax
.text:00416337  E8 0E 08 00 00    call   sub_416B4A
.text:0041633C  B9 95 06 9C A8    mov    ecx, 0A89C0695h
.text:00416341  89 0D F0 2C 41 00    mov    dword_412CF0, ecx
.text:00416347  6A 00      push    0 ; lpGQOS
.text:00416349  6A 00      push    0 ; lpSQOS
.text:0041634B  6A 00      push    0 ; lpCalleeData
.text:0041634D  6A 00      push    0 ; lpCallerData
.text:0041634F  6A 00      push    0 ; namelen
.text:00416351  6A 00      push    0 ; name
.text:00416353  6A 00      push    0 ; s
.text:00416355  FF 15 DB 20 40 00    call   ds:WSAConnect
.text:0041635B  89 05 A3 30 41 00    mov    dword_4130A3, eax
.text:00416361  81 F8 FF FF FF FF    cmp    eax, 0FFFFFFFh
.text:00416367  0F 85 4A 5D 00 00    jnz   sub_41C0B7

.text:0041636D  89 45 F8      mov    [ebp+var_8], eax
.text:00416370  FF 35 A3 30 41 00    push  dword_4130A3
.text:00416376  FF 35 F0 2C 41 00    push  dword_412CF0
.text:0041637C  53            push  ebx
.text:0041637D  6A 11        push  11h
.text:0041637F  6A 08        push  8
.text:00416381  57            push  edi
.text:00416382  FF 35 F0 2C 41 00    push  dword_412CF0
.text:00416388  6A 5A        push  5Ah ; 'Z'
.text:0041638A  FF 35 A3 30 41 00    push  dword_4130A3
.text:00416390  E8 47 1F 00 00    call   sub_4182DC
.text:00416395  89 45 E0      mov    [ebp+var_20], eax
.text:00416398  89 0B 36 DE B6    mov    ecx, 0B6DE360Bh
.text:0041639D  33 CE        xor    ecx, esi
.text:0041639F  83 E9 86      sub    ecx, 0FFFFFF86h
.text:004163A2  33 CB        xor    ecx, ebx
.text:004163A4  83 C1 58      add    ecx, 58h ; 'X'
.text:004163A7  89 4D F0      mov    [ebp+var_10], ecx
.text:004163AA  68 ED 02 41 00    push  offset GQOS ; LOPENFILENAMEA
.text:004163AF  FF 15 07 1D 40 00    call   ds:GetSaveFileNameA
.text:004163B5  89 05 A3 30 41 00    mov    dword_4130A3, eax
.text:004163BB  6A 60        push  60h ; ' '
.text:004163BD  FF 35 F0 2C 41 00    push  dword_412CF0
.text:004163C3  6A E9        push  0FFFFFFE9h
.text:004163C5  6A 09        push  9
.text:004163C7  FF 35 F0 2C 41 00    push  dword_412CF0
.text:004163CD  6A 57        push  57h ; 'W'
.text:004163CF  6A 8A        push  0FFFFFF8Ah

```

Figure 9

The other common feature is that the crypter breaks up the code into many basic blocks that are connected via jumps and return instructions. Most are unconditional jumps, but there are also ones using conditions which are always true or false — in other words, the code flow is always predefined. Further, especially at locations like non-fake API calls, the code pushes the address of the next code block or API call to the stack and uses the return instruction to jump to it. This results in spaghetti code that is difficult to decipher. Besides obfuscated disassembly code, the fake API calls can also confuse weak behaviour-based detection systems and/or may be used to detect certain emulation-based systems. Due to the amount of API calls, we haven't analyzed the latter in detail.

The next layer of protection comes with a bunch of self-modification routines. After it sets the stage hidden inside the spaghetti code, the vast majority of samples execute a self-modification routine via "call eax" (Figures 10 and 11).

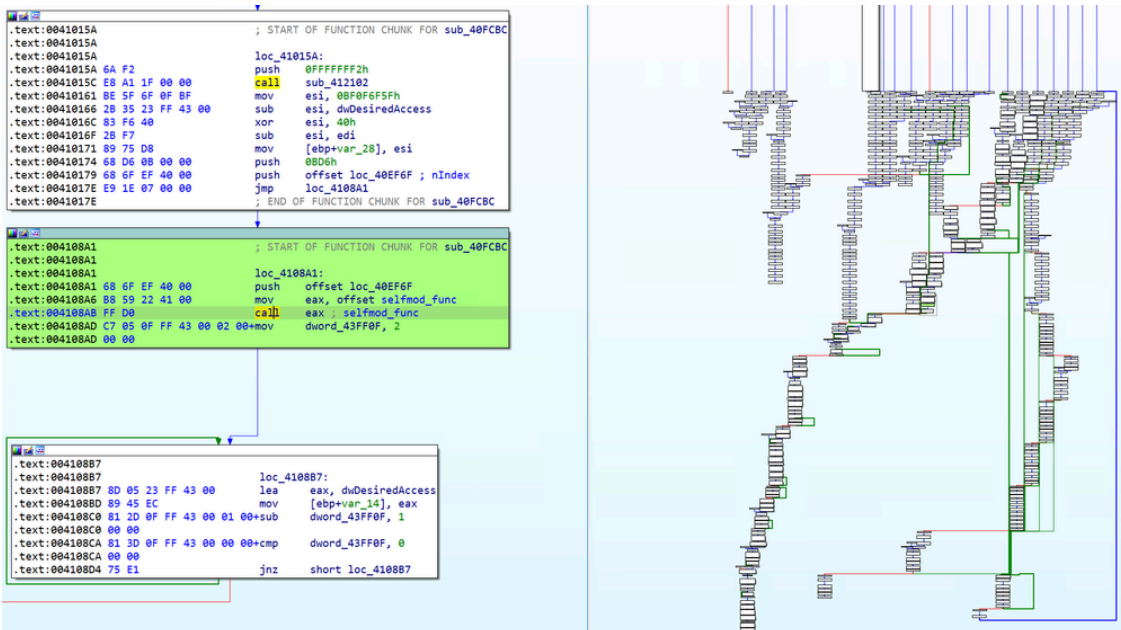


Figure 10: Sample 4bfd547775ba5892e66d2ff6a0c1de4365ab11b3\_exe.bin (946d4d332a06b9af10da38beb3e8195054840b59a870a2f9027e6471f4869dc6)

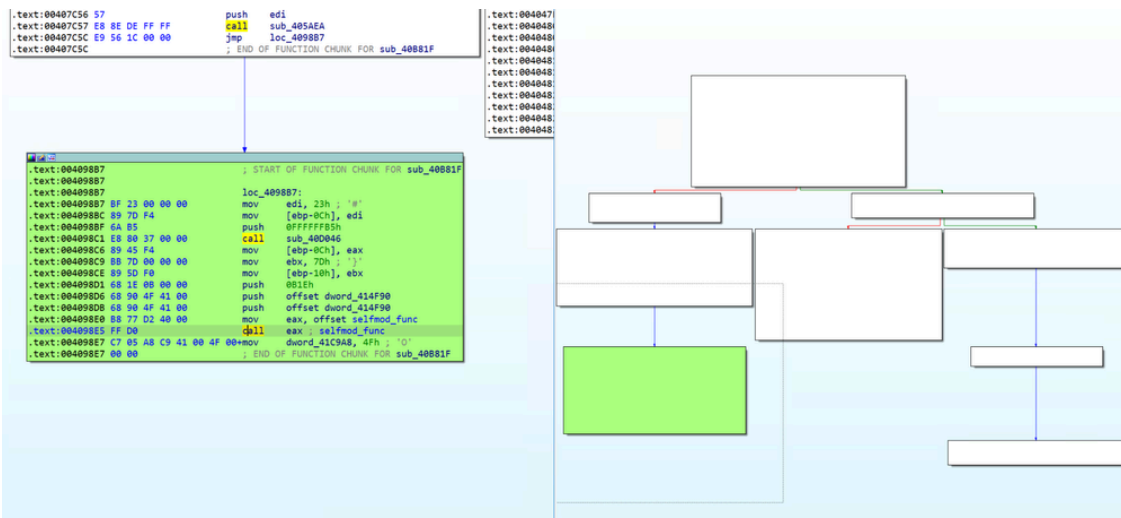


Figure 11: Sample 9679f2690d31cee38e57f080656b8618ca474c65\_exe.bin (9b28aa737bbfec90341c6a42e2d44f3308659e4fb9dd42d98a0b46cde7aaed63)

Again, as shown in Figures 12 and 13, the code around this call looks different across different samples and even the instructions used to decode the bytes which are to be modified are different across different samples.

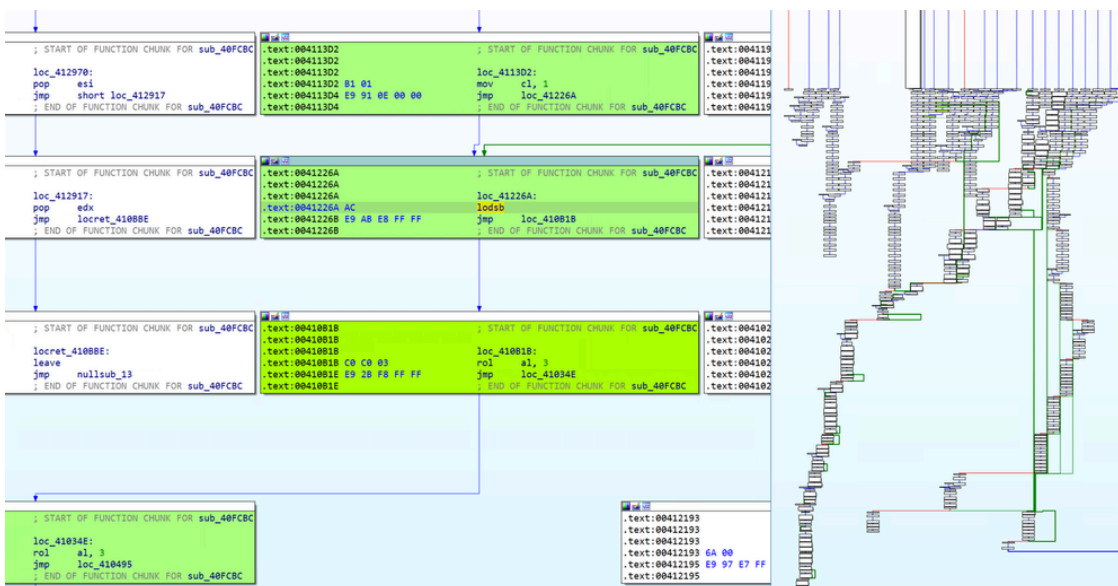


Figure 12: Sample 4bfd547775ba5892e66d2ff6a0c1de4365ab11b3\_exe.bin (946d4d332a06b9af10da38beb3e8195054840b59a870a2f9027e6471f4869dc6) decoding instructions

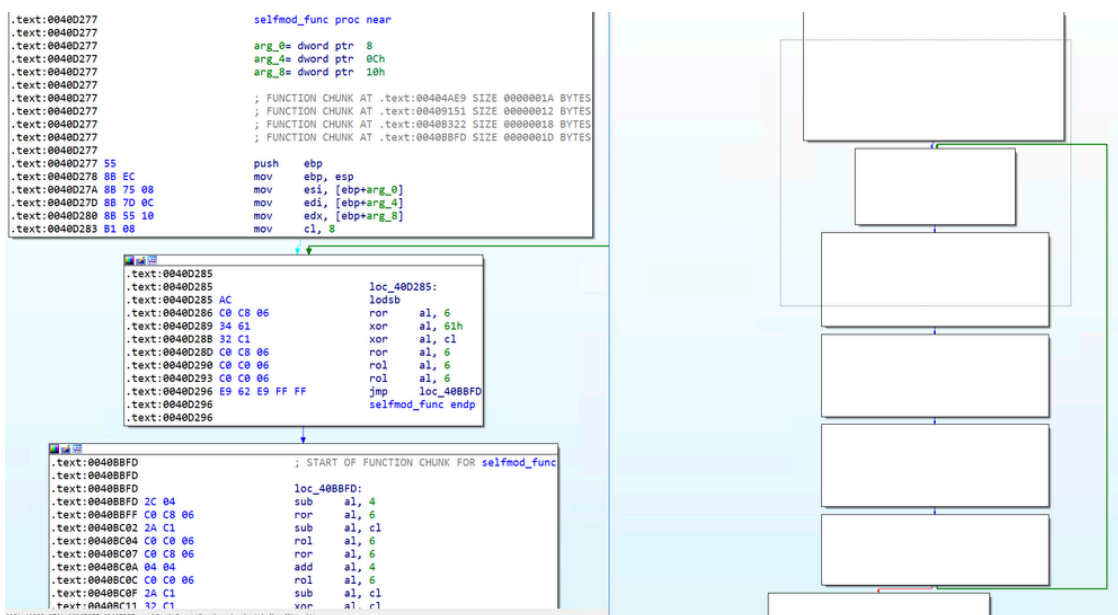


Figure 13: Sample 9679f2690d31cee38e57f080656b8618ca474c65\_exe.bin (9b28aa737bbfec90341c6a42e2d44f3308659e4fb9dd42d98a0b46cde7aaed63) decoding instructions

Tracing over the instructions in x64dbg (Figure 14) shows the differences (Figure 15) in the decoding algorithm per sample.

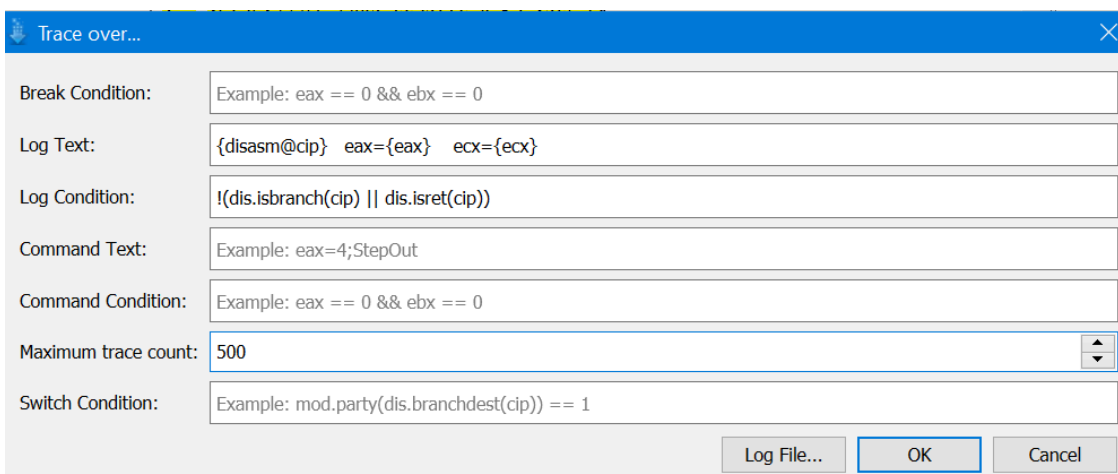


Figure 14 - X64dbg trace over settings

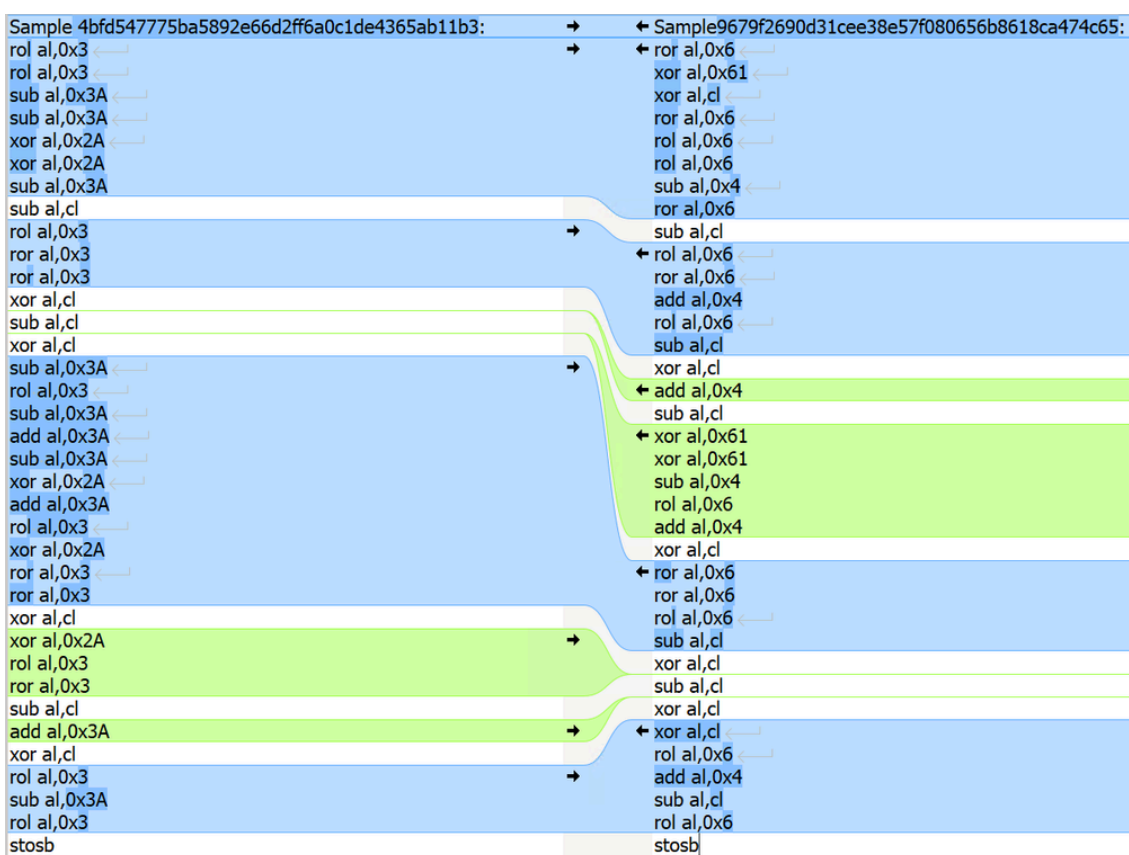


Figure 15

Looking into the decoded bytes, the first 24 bytes are just rubbish to hide the real start of the function. The function later called starts at offset 24 and looks like Figure 16.

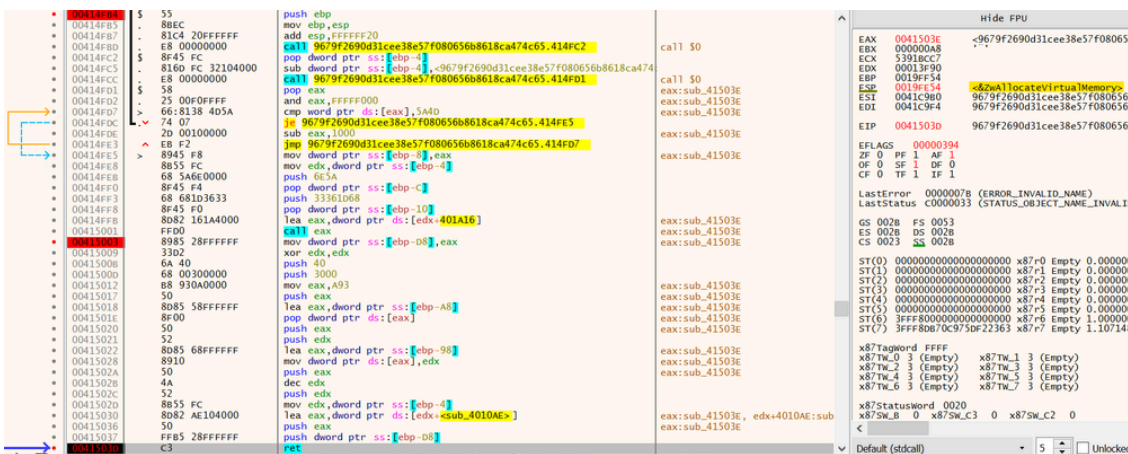


Figure 16: ZWAllocateVirtualMemory

It resolves the address of the native API call ZWAllocateVirtualMemory at address 00415003 in Figure 11 and executes it via the return instruction at 0041503D. It calls this and similar functions several times while unpacking the final payload. The number of times varies based on the sample being analyzed. It fills these allocated buffers with helper functions which are used for further self modifications or unpacking procedures of the payload. The last allocated buffer is then finally filled with the unpacked malware payload. The code used for these steps varies across samples.

Sample 9679f2690d31cee38e57f080656b8618ca474c65\_exe.bin (9b28aa737bbfec90341c6a42e2d44f3308659e4fb9dd42d98a0b46cde7aaed63) is adapting and then injecting the payload into its own PE image in memory at the end (Figure 17).

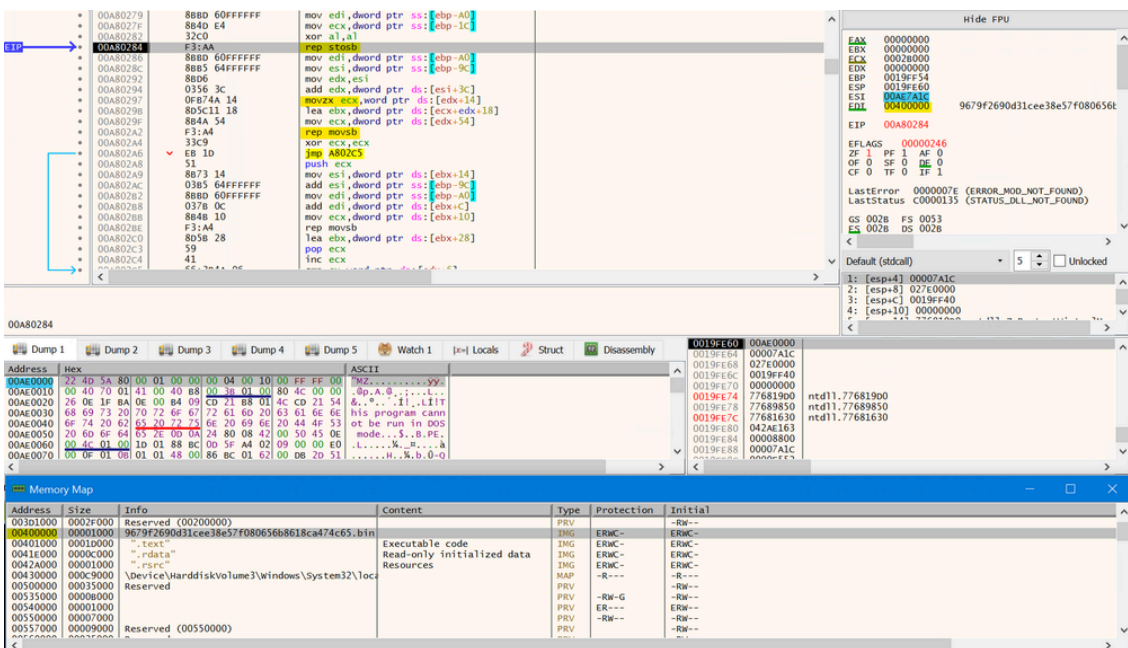


Figure 17: Overwriting own PE image in memory with payload

Other samples simply unpack the payload to the aforementioned buffer and jump into it (Figure 18) e.g. sample b95fe75736d19887ae4e3516c5cd9c7fa7caf763a138794f9aacdec8d37f0b0. At the end, no matter the sample version, they all use a "jmp edx" to hand over execution control to the payload (Figure 18 - first line).

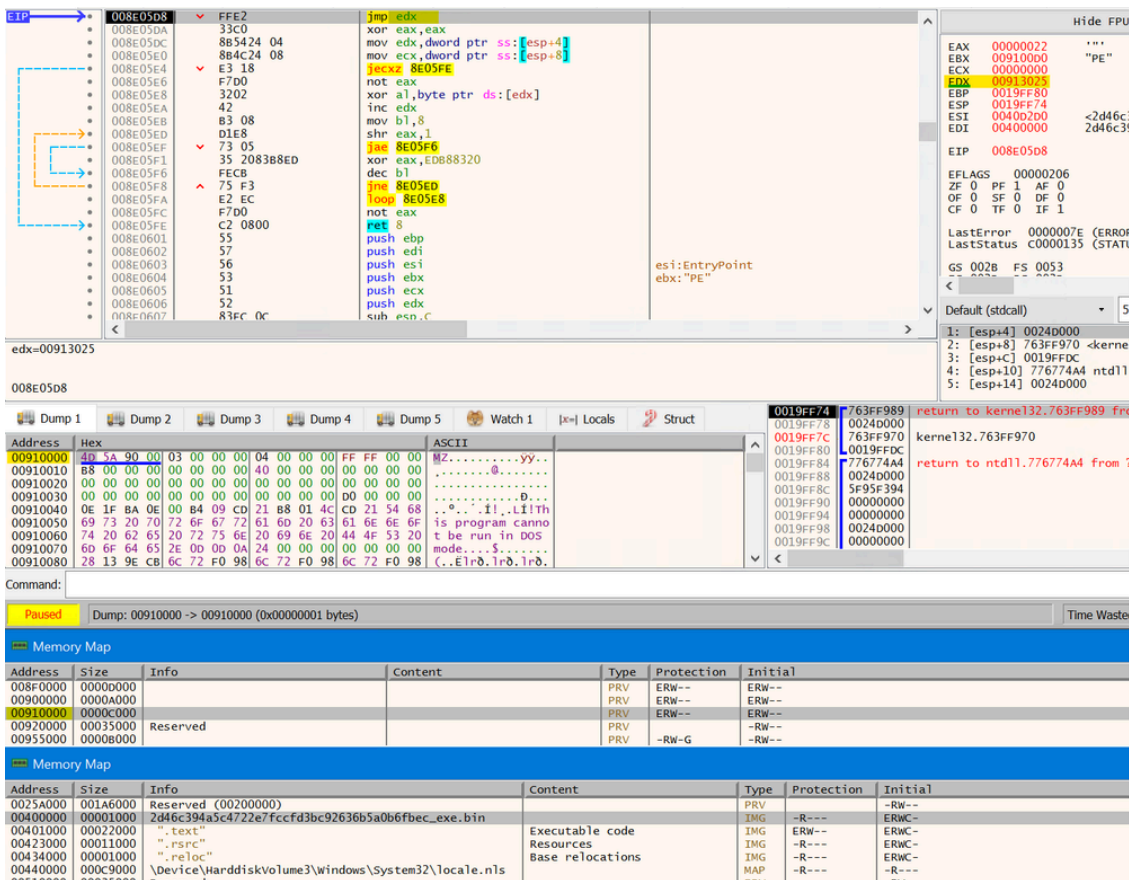


Figure 18

The last technique to mention is primarily observed in older samples, but it is hard to say if this is an evolution of the packer or just an additional feature enabled while building the packed binary.

Having a closer look to the PE image in Figures 17 and 18 also shows that the samples even have different segments. In other words, as previously mentioned, the diversity of samples packed with this crypter is quite high.

If you want to get an idea of all the different layers and self modifications we are providing two unpacking scripts for the samples 9679f2690d31cee38e57f080656b8618ca474c65\_exe.bin

(9b28aa737bbfec90341c6a42e2d44f3308659e4fb9dd42d98a0b46cde7aaed63) and

2d46c394a5c4722e7fccfd3bc92636b5a0b6fbec\_exe.bin

(b95fe75736d19887ae4e3516c5cd9c7fa7caf763a138794f9aaecdee8d37f0b0) in the Appendix of this post.

Only use these scripts for the samples mentioned in the header of the script. Be careful using them for other samples released before this blog post. Anyone looking to dissect this more should be extremely careful with new samples released after this blog post — it is very likely the adversaries will modify the packer. Keep in mind that these scripts do not work for every sample. It is recommended to single-step through the scripts (TAB) to verify that the different steps are working for the sample being analyzed.

The video below shows the unpacking process for sample

9679f2690d31cee38e57f080656b8618ca474c65\_exe.bin

(9b28aa737bbfec90341c6a42e2d44f3308659e4fb9dd42d98a0b46cde7aaed63) which is the latest sample we

looked at. The file's timestamp suggests that it was built on Monday, Aug. 24, 2020.



## Conclusion

The email distribution campaigns associated with samples built using this crypter feature several notable characteristics designed to evade detection and make tracking and analysis more difficult. The use of web-based contact forms demonstrates how adversaries can take advantage of any communications pathway to internal employees to attempt to infect organizational systems with malware. And the inclusion of legitimate hosting platforms makes it more difficult to respond to these campaigns — some organizations may require access to the same platform for legitimate business purposes. The crypter used to obfuscate various malware families features several interesting techniques which are used to make analysis more difficult. The diverse list of malware families being distributed by adversaries creates a variety of risks to organizations which should be considered by defenders who are responsible for security corporate environments. These campaigns and the refinement of the TTPs being used will likely continue for the foreseeable future

## Coverage

Ways our customers can detect and block this threat are listed below.

Product	Protection
AMP	✓
Cloudlock	N/A
CWS	✓
Email Security	N/A
Network Security	✓
Stealthwatch	N/A
Stealthwatch Cloud	N/A
Threat Grid	✓
Umbrella	✓
WSA	✓

Advanced Malware Protection ([AMP](#)) is ideally suited to prevent the execution of the malware detailed in this post. Below is a screenshot showing how AMP can protect customers from this threat. Try AMP for free [here](#).

Cisco Cloud Web Security ([CWS](#)) or Web Security Appliance ([WSA](#)) web scanning prevents access to malicious websites and detects malware used in these attacks.

Network Security appliances such as Next-Generation Firewall ([NGFW](#)), Next-Generation Intrusion Prevention System ([NGIPS](#)), and [Meraki MX](#) can detect malicious activity associated with this threat.

[Threat Grid](#) helps identify malicious binaries and build protection into all Cisco Security products.

[Umbrella](#), our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Additional protections with context to your specific environment and threat data are available from the [Firepower Management Center](#).

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on [Snort.org](#). The following SIDs have been released to detect this threat: 54920, 54921.

## Indicators of Compromise (IOCs)

The following indicators of compromise (IOCs) have been observed as being associated with attacks leveraging this malware loader.

### File Hashes (SHA256)

The following hashes are malicious binaries observed as associated with malware attacks leveraging this malware loader.

```
00272dd639402fa76db43207d074fe52d4849e5d46008f786b944a789b09afc2
00a27bcfb0940ff100f8ae4dce0eb122014926bd0541799989303a48c9327934
010c432222ef81d02edb03bfea7e866c740d4ec2f181d6ccdaa8426cf8b390c4
01453cb1e75f9a3cc2fd490c028adfcda6f63fbb63216a79379a507697dbf325e
049c5f625953b02a7aba1c904a14851cdd998ea21ee1e604016f8ba37c952ed1
051c33755b7bb0beec230c1f2af5d7880203ad2d75804719429e8054e7fb2edb
06aa2c01416bce49d24834eedd47c2dd9dc75ae39053c72168c7cc1efdd76fcd
0a6df8d51b26c7bd3a7376f6118049f1c37a91ee0b2f1ada2b86c561d1170e1b
0b61ae1202d964bea67cea9d663c8530b11c1268bcea9439b949e669b3b3be7c
0c09ea2d5722a2484804bf8c4f051d66d8a95d23c19e5d1277a55cafd80b67be
0c8569e4304f46352b041dcb692f85c9e195130db2013d4f2216130603478035
0c8fb7da34371d8c87ac3d892ceee0ba6353f00ff72d87eeebb0066d32aede56
0d261d63162d4087a82d1f67012c781cc0aaa05f8e801566f9bffa8d23981736
0d6aad4b3fe886c7d24286d333094bed9eb2c6d5ee3f7afb7fabbd1538f440e2
0dd44c11dd1b2599b8afe1121a08be81eb4230c5f1bfff335b521a74bf6baadcd
12eb2aebb455cda2819b9a1e3d18d4adcebe76ec0ee94e7676d6633965ea190f
13b155af2f1503def5c0c197902e0669faf324ec039c80a332f66f8fa4578c
1478aec44d67217a18fb2b88e9db45b9c15c468f2498f56f09a5d1fef4eafb66
161dabd778b2d24a7cae425bc2349e3db840acf49222c6067359ec7a01d3e05e
16464a294fc276ea38e4f8aedd7fa6d1f426036d42b342bde27bdc63b5c6658c
1670b07e9f5ff620a7eb773f4119023c67911655fb43bd2872021d11d05c1b0f
1be04e51510b2aa6b51598838b97124a73952c46b17d3d1c38254dd6d94e82a7
1d7680d04b5d5403b34205336ad9b9c993e6e091708da775c766e2ad5e8d8547
```

9b28aa737bbfec90341c6a42e2d44f3308659e4fb9dd42d98a0b46cde7aaed63  
1df55177a6326406d491cd9b6d4951392e565dd13e597836684468de6fec52f1  
1f5e08770d900bd0eb4f3377b05083d645e1ac9bb92f08146cd2a625363bc0c7  
20013032b298d4632abacefea5b8a6b376746c21642a8f8a38e039fd237d3c47  
201493f9e767eac36acc868f6903b574d5c9fcb78b30c5be4c1e3358e0f0026d  
2026e97bd58d8848dbd55664417790d5ee804bc2fe86ad054cb6a304d2d39a6b  
219594d5b634d5f95904376a1cbd8ecde93b8cdd6c fb785069e51e7ecc78baa  
243b7b37f5544bbbaf533f819d90b6702131fd09139d2d9d86a5a305f9ae63e  
25b24d6e39aacc3bd23e8ad26b66f6aeb29777a6038e611263dac2c586726fe7  
2b448e88df42c70da8e66df1a1c1cfe587fe9b8d5f075d3f6ae2572a7936af8f  
30ed17e5c4c4891e495af85476f5928e4d4249ee1ce305de6735e56604af2922  
32ff75054df9b2b9cefec2dcc9e4cb6c fb231bc7ad9be5a41e2d3fc37f15563b  
34f383ad792bd93bc10079ee7f2a620402f35c25655eb7f2621f4b4a0a7c74b7  
3685ce874b1abef3448d4c50ea0371e31a78afbae09287d3ff5d25953091659c  
3733c5093677fb3d4d73c7850fe399c fbf5e327ef3e151e5e279c4a9622928c5  
3f70f5fc0ef938508398fbdcd54f45cada78165fdd40d6ab95d7c2a59cad6c  
431a34f1ab6dec2c646b408b9b5ce091882244fde39498484fc0c73390d8f7f0  
431d9d2618fb3fa040f17f106617a943a43933bd19bf5cc5e4a964aabaee5674  
45763478ad7f43aa83ebaaea28bafaac6835bec45bf8c6948cf5dd454b6e30c7  
468b1785ef8bb53970a37e5399ccda3d12854587a95967081281ba857f8d2c54  
468eb27c31a7c90e3c6fc3a708dc5f78cfc5c0f8f3d76213919ca6d4e21950c1  
471325daa2bc75f50856e93e9de088386556fc3ead653894d5c2a67f2a8b4975  
47b6fae16a8c59bf1be620cb167e2673af2e0ffa92503fadf101b4bd47132ed5  
4b3ff2b99ed45563241abdf5079986e2e895ff7a011a2530e6ab9da20fa14954  
4eb476010c782e251800a7850be86b5b206ffbabe5f30bca76d20e68b13fafaa  
526abd604aacfa30338c708974493d272d8ab09df53cd1db657c3666113fbca1  
532e67f58bdc181f0df5428f72356c58a0bc fbd85b721564242983ac6e1e39bc  
549eb886be64ac9d6d2817575f5d0d0f46c0c5602fedd29aa561ab3f05e3349e  
5557fd15d615f360af1aefa6a7e2bed3382e26bdabb08d7e5a8f0f9387449f3a  
56f9b54e1e16887d66b8b9b7ea71d610951c18662a132cf7c9900d67b9745e81  
5757ebcb5b2227615a6d3a795beeff4745037313939426f9c7f8061568bbd55d  
57c7f0bc7a487df329eb3e8b19debde9ce1821e250123227cdc2dfbdd8b1b766  
5942b57d50e389ec7be01bd5b4007249e3755064fe156941dfbe310f7fa53a73  
59fc347dac3dd1c78d62393589818b5417ca041d697d155040988b14562bc797  
5b819852f844200d4d0b6ecc66451734c7421a98086d552e354709180d86d2d3  
5b831fb067dfb53992bb8a346e4fc038de6441a94ad5a3932dc8bd64f80e56fc  
5bc775300f1d06d0f92f53ec9bd8d6a919fb07101d91af05c5308063e832c34e  
5eaf5d22f937189275b6fae1257fc682194ca768a91d7b5e897770ad008f7112  
5f287d8b207645d9c fb47ff2aacb7ed2a6769fa14b1fe78c45a73efc73f0a84c  
614dc6fb2ee570cc9d1782534a64a3ea2c0be9d213e806141bb28d23542f6c47  
6297c5e9a01f95d14ab4928e69bbc0f7cc4113b08f782e5ea9f04b1d1814b74b  
630c098ef8211c05e0e68008bcbcd6e4402e580e9538817b56084dc301954426  
64965473405235f37a721d9fecaf4ccb0ca038a3014538eba80cd9943beb1d7  
6627f0ea1f8ac35ea8567a1677ed4c67e7429344a56a80a094706fcd6d063221  
6b2e0ce03329ccb773c3c31f68ea9a74991cd0445f9e69df3c8e01ab6498d2d5  
6ce892674ac5a9607abafaf1011210cf9cb8bec2dc683b8dbf3c71c5e75a47fd  
6d3ff9a1f4745625b44f5361e4d8d78b38898d24edda328c2448a2bb377c99a

6f21364272988368a75692e2e30970ce18e625bd7b2bcb154353415b31816d4e  
6fee95a3e283d9ad09a399e99b086ef70c3679eb8ad548161bdfababe3da68fb  
725e1b82b882d693028f13e3a082a8a9f278b79ad5ade994ccc9414b4e46d32c  
72a744dcfd609870d53a513083d89a75c41c87d178025821963a1353216d0090  
72f7451a21dcf26ebba525994d312a06483ef3fda0ffcd6500e36b94dcf78681  
7556a3160ba28ac50418bb9989de2f83a5dfc54376a43500d16db7c3c76c04d4  
76776877cf663f9b3d5647b0efd3e062cf41d7d4757371ed9ad4fc0d85d7179  
77dd0d459a930c4e2692f935d775f9aa6560e26b19715ee61c20ce6bdbcc8200  
7da2c8166c73f7f5cf4f8f07b3ad161fe7cac6aa87c6a2701cb7fdc74242dfa6  
827fb4f521e8f22fb91ee61d921d3287bab18c9f01d333139e4124c989d34c76  
8281d87c1343eed7d48c8d98043864ace2f0dade93628cbda7ebc2eb07c77dd7  
87c04cc74ea6e8958bcb6319f7c1f8293d63ce4a2e34a3c99ba296e55fd0ff4  
88c23ddbc21c130588f88500c753138e1b03a55f4658ff199c247fd8ceee5bd3  
890eacb1a49d606586eb585ee0738f55ac76fb3a175016ad627532425ee19dc8  
8a237182974d55a414f91a6d657403fafc8b79685ff1a73562758b333aeea590  
8be791cfaf6dbe2f1022406cbab97c3f53a084abd5b7e2ede043bd10de268352  
915f41dbb8a500f22f5aa346a0b6eff9db5fb6149b936968476ad585694688c5  
91f7787447cdaaec0f0feb10440bc2cb2c48ec8f3ce96a49719cbb9a20b3667d  
91f9f150e8a010ac25cdfd1a5489dd87f5e4af1d1daf8d12710748fb49396f67  
932b193e475a885c612b6610ad40177e65fc18eba76485e7da690cd4c9c5894a  
942331c18f6435087cc15c1012f4b4c9c7265d6ffaef0650aad6b6ce4064c5c3c  
9444adaa9f6597a02f2e7f889b65db228c489658e0a81c28a3ac2d51f88b5e3c  
946d4d332a06b9af10da38beb3e8195054840b59a870a2f9027e6471f4869dc6  
98c59ba70665df6ff4bbabbad56b11ea24e1bb478fd0547a02dbfbacd6bbfdf4  
99fe4effb3bd5d31a6c9b740f693460042e9b327b6e6d78aaa04d40762784c8f  
9a6b7c1ebfe3ba98cbfc7d3cdd67fdb34dcf86b411eb7502c3e6d865084742b6  
9c2eb5790e7874950db0e51764ba68bb4a0fa5be68b659717c73363883da0db7  
9ff5246727eb8baf06d825830b061df21a7e7b8ecf0f4f584da3b886643f84ed  
a2387ef5d3af113c8c902f478df1c2d7f7a7acf729873b13508c1f1915bf5000  
a27f8297924f2318e1e4d2da534be65806198a6e9d53c905a79ed3f93f5c52a9  
a2e7f64c3d584a25a35bdedec0d394033a2ebb54d3457e09ceea3a623c208491  
a654dbed7f65e6e6b8830e42b137fd85ddce872dc29ca006e8f7653f0aadca7  
a66f6d5fe714527cc94af30695cbabc44dd2fc355bc8e917e77350f35b0c6852  
a770fcb1791d4a1c1037921403f936fb699b4e86df500abc41cb7fa1d762e302  
a7b83a87772511641557b1ace41c478ecf6f1be0e1585cf6ce170cbaab16d6bb  
a8231bdd44d469fd1a30464e238d0efa7c257566f301dae2fd80bdc43893b5d2  
aa33731aa48e2ea6d1eaab7c425f9001182c0e73e0226eb01145d6b78d7cb9eb  
aaa7efb4c15a71baff014071a71aa08f17e71c433016b2a0f05caac5f35adbe5  
ab24760ec387919fc8d177510f39f36ac0ee011f8349faf63456e83aa0e50944  
ad2b0478c0c3c717f6dabb96a6e957ea000462a8f8792f1c73868f9e894e037a  
adce8f29cb6cbbc44f5566031370753da43ea431ad3aeb11a6fc2186d571c957  
b1c6c7afedd5fdf67d7e9bbec952945e31c136785de3bc33313aede44f1696cf  
b240ff47de5859d0428c980a30a36a787e141761e8d7c8b2db8ab2364d226a94  
b29c4a567b439ef6072e41c38091c0f8d4d2bc0955d60062169f35e4a1cf9de4  
b3cb381f114ec9c163335509a217bbe1c6baa8d2cf5655b5ff84fd8d0a28dc9a  
b6a02bdf54b97688fda5137215e5237bd9df650bb052094dbc37947b59a93122  
b71a38ed297ef72a77fbbefa8f326a0c752f0f4fb5d80d1bf8e393824e99223

b8e75cbb6e4c360fec71e12b85f3c5e4933e66a64928ff4d337ebe6a6cc9150f  
b95349e88b1c800c3cb84bd3875aa556c7107900567b9c9fb54047bafa364dc1  
b95fe75736d19887ae4e3516c5cd9c7fa7caf763a138794f9aaecdee8d37f0b0  
ba5fa7cc1a918b866354f4a5d9d2ceb3965ff81eb96e1608f190bccf12d38e6  
bb33e0b91b1b38802b2bfd5654d3888dee17959c6f07d3ce67dbac278712a402  
bc275cd76478e4d3387740dd955d9b9b5b36f064656ecb1e1cea9b8649eec57d  
bc2c5e2692df4493a1bbc9364689f1b7c532964497512065d3693cae50214860  
bcf19173facde547cd07eb8495bd48e8eea67c4f2441d4239e883f264744de7e  
c12c749cbb465484968bf99c46a51ce717620dde2a5cd9b0168b104b604ba42  
c1665f6717238ba405059e44533e1fb7ad35798bb3d8b6ebb29cf0428241d721  
c50b34da0b0d1cc4728e4af9ae2c921df26ef71869bba0d5f9a24caa229529c9  
c679503d8fe5e6b127af12142f8a3ae32e94ef0751337de67c9dcb4dea180987  
c772f069931f2c0429bea0919fd7238de92618e6bea9e6581129656f6d86170c  
ca4214e15181e52923f713771455f5709e4baa13626cad85b735734cf66d36db  
ca9d4a7488644ce5fa20248e388291d60ca4a3bfe783acae4f02821618d1563f  
ccb8e17d3df37549a3dbdb31f3b5e03b8fbb3ddf9462cf962518c8e0312cdb56  
cda685d0e85b2f28c00d145da358edc3748357786455965af86a3b0b292051fd  
ce30c658a17cdced008ca04ea67f5ddb4c3c8d9042fe700ea89731c3d631628b  
d175b76aae5c16a5870df88cadcd1d147c78a4977cc8dd4b213224e031cbcb7fb  
d3255ed380e290f4992701d1c10a3f65580b5e0aff384ab4308a8202d71f38a8  
d3639a02c300ab5f1063c7000fc8fa9eda8b69ac98840e5e49e26ea578274c4d  
d4d3cb3f7ffed9e1847a3b47a41d1da3c649374cd1d4772baaa95e62735ac3ca  
d7bcc39c42e66f4d7ce42d49aadbd63ac541e39889dc72ee41d140c919a92d85  
da5d88c3090b95ab3bca96669880af7a39b7d8ce9a206ec1bb558086ccacff41  
de5947af1ec1e205ed4c953359a8ca9b6a72dc4ffc81ce1048969553d440c265  
deb8e521c4509a83b61d96291efe3cdaf2379c953f0590433d98a0bbd6bc82ab  
df638bde1ffaafc7b7a25fe30b631d549299f9502f23efc47fc5efaa118e96b97  
e0b15c5b68a21db7a86f92689c5df63d343a52e3e7b09d19f3ffbf941b32c4d1  
e0f5471baf3e0e4b3eb3711f89479a910fe68b75ad53070c58302821ddb84220  
e38c9a5d62aec8a807336ef40668f82a9bf764a8102fe464ee7f82041e007bee  
e5a6108aa4f8be08c8192590ab2b3231e9a33277ead76a0420557258220383be  
e655f1afb49be062cdded5683df9292ff4cd602ad5d6f648ddd8778af13c44e2  
e6b30d246362081d7cb76a4d21584f6dbccff535812aed8e9675c17ca518c0242  
eafb21a0f9f082fa2e94e010569d7fa8512d978087c2633d652b865b922465a  
ec1efcb050ec701c4a5895215e8a5ff9a89faf5b56976975de4e3846577841f0  
ee42a34b83f4c27c57ebfa79f78d4702cdf5c845443b929b9e9d3246409aacca  
f1d43050a57365d41f2ec6ac3f73f1ddf9fa454c8b96aa5aa30749490b5a5fd4  
f29484e8e60d1104bb60a504279909f7d4c5a68d77d794dcaed4758d95989a99  
f38a38d97bd96a42e8c3f985f1e65daca97dce229b36c8c80c9229666826f325  
f431b5b6dc04208fe9ff89eafd94c92d1383857d17702240f14aa19f538b3aa  
f6c96250359377ca85340b8c2c7253dd4f8fb5b1b8bf87d9fbce45ae40fe5417  
f6ebd6f0fe20fe561d1c5fd6aea5201712a0eabf4624c863a5ab6d44b1f57755  
fc856c48ddb5be64f343a482faace641542c06e9a52556c16a8a80f973347ad  
fd03a9145d89c102799e5719608a28da681c1289352eb1d40b557cc667ba2ad0  
fd95fc2bc6b3362454dc6a0f0aa03f8c481ece2442e544b4ea0d9b0f81cd8b8c

fe0d4a9ac1d0e3a626b44357e4469f402b9dad3f020776ecf771da693a782d61  
ff5fc5c5318fa051992c7c3408d203f306c13b5fcd9400f860f734ce47a3b676

## Domains

The following domains have been observed during various portions of the malware infection process.

123pcloud[.]com  
92g938uextmgvb7rllv8wcad[.]biz  
98iudjsandsas[.]info  
df1[.]kamalak[.]at  
aes[.]one  
amfibiyapolyakova[.]com  
api3[.]lamanak[.]at  
askyourspace[.]com  
banisdor[.]top  
banudarog[.]com  
banusle[.]top  
basa[.]nutarborg[.]com  
bnxhbc25hwc8b8afawhevzw[.]biz  
calmstill[.]xyz  
cheneer[.]org  
choksaiiwkoksall[.]info  
cmck4vve6e24wdktsc9n8l1izi7eb9[.]biz  
crocopexpire[.]ug  
cromecho[.]com  
cupersip[.]com  
dailystepstowardsuccess[.]com  
dasifosafjasfhasf[.]com  
deliverynice[.]club  
dksadjsahnfaskmsa[.]com  
dksdjasi92iejdnfsa[.]info  
dksjdsajdiei28uj2[.]info  
dodontrami[.]com  
drysetfirst[.]com  
dsdjfh9ddksaas[.]pro  
dsdjfhdsufudhjas[.]com  
dsdjfhdsufudhjas[.]pro  
dsjdjsadsadhasdas[.]com  
dskdsajdsadasda[.]info  
dskdsajdsahda[.]info  
dskjdsadhsahjsas[.]info  
dwajfjaiakdnsandks[.]com  
dweandro[.]com  
esplody[.]org  
fastandstrongwolf[.]com

fdsjffjdsfjdsdsajjs[.]com  
fdsjffjdsfjdsjfdjsfh[.]com  
fedex-tracking[.]press  
findulz[.]com  
fredoam[.]com  
fslakdasjdnsasjsj[.]com  
giridly[.]com  
gstat[.]rayzacastillo[.]com  
haponebitold[.]com  
heclinggotof[.]com  
idisaudhasdhasdj[.]com  
idsakjfsanfaskj[.]com  
iloveyoubaby1[.]pro  
infinitydeveloperspes[.]info  
informatioshopname[.]ru  
ipnfbqg2raz3asn4j631ha453bbr4h[.]biz  
islacangrejo[.]fun  
j2888hennene[.]site  
jdafiasfjsafahhfs[.]com  
kasfajfsafhasfhaf[.]com  
kdsidsiadsakfsas[.]com  
line[.]largefamiliesonpurpose[.]com  
line[.]lawnteam[.]org  
line[.]monalisapizzerias[.]com  
line[.]rllconsulting[.]com  
line[.]zepcnc[.]com  
link[.]paichecafe[.]com  
link[.]philippeschellekens[.]com  
manutobis[.]top  
menosita[.]top  
mesoplano[.]com  
morenodorf[.]com  
mifastubiv[.]ru  
morentok[.]top  
oajdasndkdahm[.]com  
odoncrol[.]com  
oi2jidsdjsdd[.]info  
opetileon[.]ru  
orderrys[.]com  
paiancil[.]com  
petronasconn[.]ru  
phanleb[.]com  
pics[.]crystalridgedesigns[.]com  
pleclep[.]com  
procinul[.]com  
redsobabtert[.]com  
retordownty[.]com

```
rolhorabdidn[.]ru  
rygotunren[.]ru  
service[.]pandtelectric[.]com  
shoolman[.]ca  
siciliyaopartion[.]ru  
sl9xa73g7u3eo07wt42n7f4vin5fzh[.]biz  
smarteyecare[.]in  
soletrobuse[.]ru  
staycalm[.]club  
stoutorder[.]xyz  
sweleger[.]com  
syndicationtwimg[.]site  
telete[.]in  
teoresp[.]com  
toptopcoop[.]info  
toptopcop[.]info  
twitter[.]website  
ukronet[.]ru  
unverifiedintigoosjai[.]info  
weksrubaz[.]ru  
wp[.]quercus[.]palustris[.]dk  
wunchilm[.]com  
yamaha[.]ug  
zonculet[.]com
```

## IP Addresses

The following IP addresses have been observed as being infrastructure used for delivery and post-compromise network communications as part of these malware campaigns.

```
109[.]248[.]111[.]134  
109[.]94[.]209[.]7  
139[.]60[.]161[.]58  
141[.]255[.]166[.]149  
142[.]93[.]110[.]250  
166[.]62[.]30[.]148  
179[.]43[.]147[.]73  
185[.]153[.]196[.]209  
185[.]158[.]249[.]63  
185[.]252[.]144[.]191  
188[.]127[.]230[.]211  
192[.]155[.]111[.]215  
193[.]38[.]55[.]23  
193[.]38[.]55[.]92  
194[.]67[.]78[.]65  
195[.]123[.]209[.]4
```

```
195[.]123[.]214[.]163
195[.]140[.]164[.]58
195[.]201[.]225[.]248
31[.]148[.]99[.]73
31[.]184[.]253[.]171
31[.]184[.]253[.]248
31[.]44[.]184[.]125
31[.]44[.]184[.]50
34[.]240[.]96[.]52
40[.]81[.]188[.]85
45[.]84[.]227[.]231
46[.]20[.]33[.]219
47[.]241[.]8[.]147
47[.]57[.]89[.]207
5[.]101[.]51[.]172
5[.]188[.]62[.]165
5[.]63[.]159[.]168
62[.]108[.]35[.]53
78[.]46[.]233[.]14
80[.]249[.]144[.]38
80[.]78[.]254[.]167
84[.]38[.]181[.]209
84[.]38[.]183[.]147
84[.]38[.]183[.]162
84[.]38[.]183[.]181
85[.]143[.]222[.]85
88[.]80[.]186[.]83
```

## Appendix

### X64dbg Unpacking scripts:

#### Script 1:

```
// Unpacker script for sample 2d46c394a5c4722e7fccfd3bc92636b5a0b6fbec_exe.bin
// (b95fe75736d19887ae4e3516c5cd9c7fa7caf763a138794f9aaecdee8d37f0b0)
//
// Author: Holger Unterbrink (twitter: hunterbr72)
//
// This script should show the logic of the packer, it is not an optimized unpacking script
//
// Load sample and go to entry point, then execute script:
// (there are some race conditions, single step (TAB) through the script)
//
//
// find first call eax
```

```
TraceOverConditional ReadByte(cip)==FF&&ReadByte(cip+1)==D0
// unpacked code stored in [csp], offset 24 begin of code, first 24 bytes are only used for obfuscation
mov $code, [csp] + 24
log "Selfmodded code at: {0}", $code
StepOver
bp $code
run
// run until return
TraceOverConditional dis.isret(cip)
//pause
bc
StepInto
mov $codeptr, [csp+8]
bp cip
run
// from here some samples are different, some allocate more buffers, some less
run
run
run
mov $codeptr, [csp+8]
StepOver
StepOver
StepOver
StepInto
mov $code, [$codeptr]
bc
SetHardwareBreakpoint $code,r,1
log "Final payload at: {0}", $code
run
bphc
// run until jmp edx - OEP unpacked payload
TraceOverConditional ReadByte(cip)==FF&&ReadByte(cip+1)==E2
log "Dump payload malware at: {0}", $code
ret
```

## Script 2:

```
// Unpacking script for sample 9679f2690d31cee38e57f080656b8618ca474c65_exe.bin
// (9b28aa737bbfec90341c6a42e2d44f3308659e4fb9dd42d98a0b46cde7aaed63)
//
// Author: Holger Unterbrink (twitter: hunterbr72)
//
// This script should show the logic of the packer, it is not an optimized unpacking script
//
// Load sample and go to entry point, then execute script:
// (there are some race conditions, single step (TAB) through the script)
```

```
//  
//  
// find first call eax  
TraceOverConditional ReadByte(cip)==FF&&ReadByte(cip+1)==D0  
// unpacked code stored in [csp], offset 24 begin of code, first 24 bytes are only used for obfuscation  
mov $code, [csp] + 24  
log "Selfmodded code at: {0}", $code  
StepOver  
bp $code  
run  
TraceOverConditional dis.isret(cip)  
//pause  
bc  
StepInto  
// set breakpoint at <ntdll.ZwAllocateVirtualMemory>  
bp cip  
// save ptr to allocated memory  
mov $bufferptr, [csp+8]  
StepOver  
StepOver  
StepOver  
StepOver  
mov $code1, [$bufferptr]  
// change to Dump 1  
dump $code1  
SetHardwareBreakpoint $code1,r,1  
// this takes a while, only hit tab once, be patient  
run  
bphc  
// Dump1 get filled  
StepOver  
StepOver  
// unpacked and jmp to memory alloc (ZwAllocateVirtualMemory) function  
run  
// break at ZwAllocateVirtualMemory, set bp to allocate buffer  
mov $bufferptr, [csp+8]  
StepOver  
StepOver  
StepOver  
StepOver  
mov $code2, [$bufferptr]  
// change to dump2  
dump $code2  
SetHardwareBreakpoint $code2,r,1  
run  
// break at ZwAllocateVirtualMemory, set bp to allocated buffer  
mov $bufferptr, [csp+8]
```

```
StepOver
StepOver
StepOver
StepOver
mov $code3, [$bufferptr]
// change to dump3
dump $code3
SetHardwareBreakpoint $code3,r,1
run
// Dump2 get filled
StepOver
// you can keep the code2 bp if you want to see the decoding of the buffer
bphc
// re-enable code3 bp
SetHardwareBreakpoint $code3,r,1
run
bphc
// code buffer3 is filled (1st part of decoded payload)
TraceOverConditional dis.isret(cip)
TraceOverConditional dis.isret(cip)
TraceOverConditional dis.isret(cip)
// first part of payload PE unpacked, further fixing PE
StepOver
// ZwProtectVirtualMemory changing 0019FF3C to ERW
TraceOverConditional dis.isret(cip)
// overwrite org PE image in memory at 0x400000 Len 0x2B00
TraceOverConditional ReadByte(cip)==F3&&ReadByte(cip+1)==AA
// fix org. PE header (first 200Byte)
TraceOverConditional ReadByte(cip)==F3&&ReadByte(cip+1)==A4
bc
StepOver
TraceOverConditional ReadByte(cip)==FF&&ReadByte(cip+1)==E2
// hand over control flow to payload malware
log "Payload OEP: {edx}"
```

---

Source: <https://blog.talosintelligence.com/2020/09/salfram-robbing-place-without-removing.html>