

AbaddonPOS: A new point of sale threat linked to Vawtrak | Proofpoint US

By November 11, 2015 Darien Huss

Published: 2015-11-11 · Archived: 2026-04-05 12:58:12 UTC

UPDATED 11/24/2015

Point of sale (PoS) malware has been implicated in some of the biggest recent data breaches, striking retailers, restaurants, hospitality and organizations from a variety of industries, and often targeting consumers in the US. [1] Once considered too difficult to carry out to be practical for cybercriminals, the retail breaches of late 2013 demonstrated that these attacks are both feasible and highly profitable for cybercriminals, and PoS malware has since continued to evolve and grow in both variety and sophistication. [2]

Proofpoint threat researchers recently detected a new addition to PoS malware landscape. Named AbaddonPOS by Proofpoint researchers, this sample was initially discovered as it was being downloaded in the process of a Vawtrak infection. This use of additional payloads to enhance attack capabilities offers another example of efforts by threat actors to expand their target surfaces through the delivery of multiple payloads in a single campaign, in this case by including potential PoS terminals. This post will analyze AbaddonPOS; discuss the observed infection vectors; and expose, details on the downloader used to retrieve this new PoS malware. We will also provide evidence to demonstrate that the downloader malware and PoS malware are closely related, perhaps even written by the same actor or actors.

Known infection vectors

On October 8, Proofpoint researchers observed Vawtrak [3] (project ID 5) downloading TinyLoader, a downloader that uses a custom protocol for downloading executable payloads from its command and control (C2) server. TinyLoader was then used to download another downloader in the form of shellcode, which then downloaded AbaddonPOS. Although this infection vector was initially specific to Vawtrak’s project ID 5, we have also since observed it delivered in project IDs 6, 9, 10, 12, and 13. The project ID’s are most easily observed with Vawtrak C2 traffic, as they are stored encoded in the PHPSESSID cookie value. Using the cookie value we provided as an example in our research on Vawtrak enables us to see it in a decoded state (Fig. 1). Bytes 4-7 contain the project ID in little-endian byte order.

```
Encoded Cookie: 5C8EC19E61666B717F808B939EA4B6C5
Decoded Cookie:
00000000: 5C D2 9C C1 03 00 00 00 07 00 02 00 00 0E 01 00  \.....
Campaign: 3
```

Figure 1: Decoded Vawtrak cookie displaying campaign/project ID

In addition to observing AbaddonPOS as it was delivered by an Angler EK → Bedep → Vawtrak infection (Cyphort, [4]) and Angler EK → Bedep (bypassing Vawtrak), Proofpoint researchers have also observed this infection behavior delivered by weaponized Microsoft[®] Office documents downloading Pony → Vawtrak (Fig. 2).

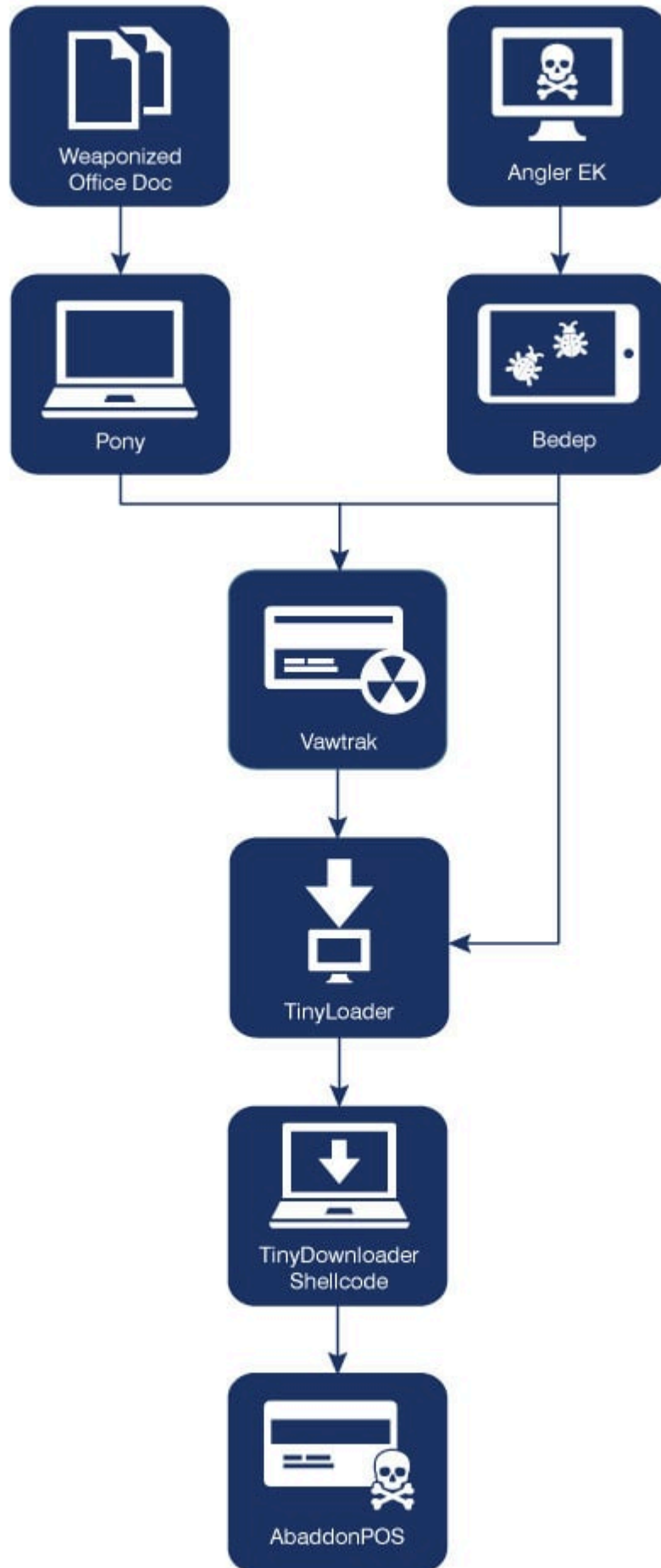


Figure 2: AbaddonPOS infection chain

TinyLoader

TinyLoader’s sole purpose in this infection chain is to retrieve executable instructions from the C2, which allows the attackers to execute their own custom shellcode on infected machines in addition to downloading and executing additional malware payloads. True to its name, TinyLoader is typically 2-5KB in size. One notable characteristic of TinyLoader is that prior to contacting its single hardcoded C2 IP address, the malware will first check to see if it is running as an x64 or x86 process using the IsWow64Process Windows API (Fig 3.). TinyLoader selects a value based on the result of this API call, and the result is then used to tell the C2 which executable code should be downloaded to the infected client.

```

002D01E4 FF95 B0020000 CALL DWORD PTR SS:[EBP+2B01] kernel32.IsWow64Process
002D01E8 83B0 40050000 CMP DWORD PTR SS:[EBP+540],1
002D01F1 75 02 JNE SHORT 002D01F5
002D01F3 EB 0C JMP SHORT 002D0201
002D01F5 C785 40050000 MOV DWORD PTR SS:[EBP+540],84000000 x86
002D01FF EB 0A JMP SHORT 002D020B
002D0201 C785 40050000 MOV DWORD PTR SS:[EBP+540],BA000000 x64
    
```

Figure 3: TinyLoader API call checking for x86 or x64

As shown in Figure 3 above, 0x84 is used with x86 processes while 0xBA is used with x64 processes; however, the values used for each architecture vary depending on the variant. Once the correct architecture is selected, TinyLoader builds a packet to send to the C2 to initiate the payload download process. Prior to retrieving the downloader that downloads AbaddonPOS, we have observed TinyLoader first retrieve a copy of itself (this step may vary slightly), which is then used as a persistence method by adding a registry key to HKLMSOFTWARE\Microsoft\Windows\CurrentVersion\Run (Fig. 4). TinyLoader may also download a DLL version of itself, in which case the registry key observed is similar to the following: regsvr32.exe /s “C:\PROGRA~2[a-zA-Z0-9]+\dll”

Name	Type	Data
(Default)	REG_SZ	(value not set)
vssadmin	REG_SZ	"C:\PROGRA~2\vssadmin.exe"

Figure 4: Example of TinyLoader persistence registry key

Once the persistent payload is written to disk, another payload is downloaded by TinyLoader in the form of shellcode (Fig. 5), the purpose of which is to manually craft a HTTP request that is then used to download an AbaddonPOS payload (Fig. 6).

case, the CALL instruction is used to push the address containing a string (Fig. 7): specifically, the address containing the string “devil_host” is pushed onto the stack, which is then used as a mutex.

```

00401BD5 | > E8 0B000000 | CALL 00401BE5 | Name = "j", jump over immediate data
00401BDA | > 64 65 76 69 6C 5F | ASCII "devil_host",0 | ASCII "devil_host"
00401BE5 | > 6A 00 | PUSH 0 | InitialOwner = FALSE
00401BE7 | > 6A 00 | PUSH 0 | pSecurity = NULL
00401BE9 | > FF15 E0204000 | CALL DWORD PTR DS:[<&kernel32.CreateMut | KERNEL32.CreateMutexA
    
```

Figure 7: AbaddonPOS using CALL instruction to hinder static analysis

Most of AbaddonPOS’ code is not obfuscated or packed, with the exception of the code used to encode and transmit stolen credit card data. This shellcode is encoded using a 4-byte XOR key; however the key is not hardcoded. Instead, using the first 4-bytes of the decoded shellcode, the malware iterates over all possible 4-byte XOR keys until the correct one is found by checking the result against the hardcoded instructions: 0x5589E58B (Fig. 8). Once the XOR result matches the hardcoded instructions, then the correct key has been found and the malware continues to decode the shellcode using that key.

```

00401D7B | > 310B | XOR DWORD PTR DS:[EBX],ECX
00401D7D | > 813B 5589E58B | CMP DWORD PTR DS:[EBX],8BE58955
00401D83 | > 74 0E | JE SHORT 00401D93
00401D85 | > 83F8 00 | CMP EAX,0
00401D88 | > 75 09 | JNE SHORT 00401D93
00401D8A | > 310B | XOR DWORD PTR DS:[EBX],ECX
00401D8C | > 29C3 | SUB EBX,EAX
00401D8E | > 31C0 | XOR EAX,EAX
00401D90 | > 41 | INC ECX
00401D91 | > EB E8 | JMP SHORT 00401D7B
00401D93 | > 3D 5C010000 | CMP EAX,15C
00401D98 | > 73 08 | JAE SHORT 00401DA2
00401D9A | > 83C0 04 | ADD EAX,4
00401D9D | > 83C3 04 | ADD EBX,4
00401DA0 | > EB D9 | JMP SHORT 00401D7B
    
```

Figure 8: AbaddonPOS shellcode decoding routine

Locating credit card data

AbaddonPOS searches for credit cards by reading the memory of all processes except itself by first blacklisting its own PID using the GetCurrentProcessId API. To find credit card data, AbaddonPOS roughly follows this process:

1. Search for 3, 4, 5, or 6 string characters, indicating the first number of a potential credit card
2. Credit card number length >= 13 and <= 19
3. Valid track delimiter (track 1: “^”, track 2: “=”, or “D”)
4. Track 1 max length: 120, Track 2 max length: 60
5. Additional checks based on whether track 1 or track 2 delimiters were found
6. Check credit card number with the Luhn algorithm

The AbaddonPOS sample with md5 hash: f63e0a7ca8349e02342c502157ec485d was analyzed for the process above. The slightly older version of AbaddonPOS may contain slightly modified functionality, including not allowing “D” as a track 2 delimiter.

Exfiltrating stolen credit card data

Although many of the different PoS malware families rely on HTTP to exfiltrate data, AbaddonPOS uses a custom binary protocol. Communication and exfiltration of credit card data is carried out by the decoded shellcode discussed above. A single hardcoded IP address is used as the C2 address, as well as the encoding routine that is used to obfuscate exfiltrated data. An example of the network traffic generated during a single credit card data

exfiltration attempt is shown in Figure 9. As a result of this analysis, Proofpoint created and published ET Pro IDPS signatures (ID's 2814677-2814680) to detect exfiltration attempts on October 30.

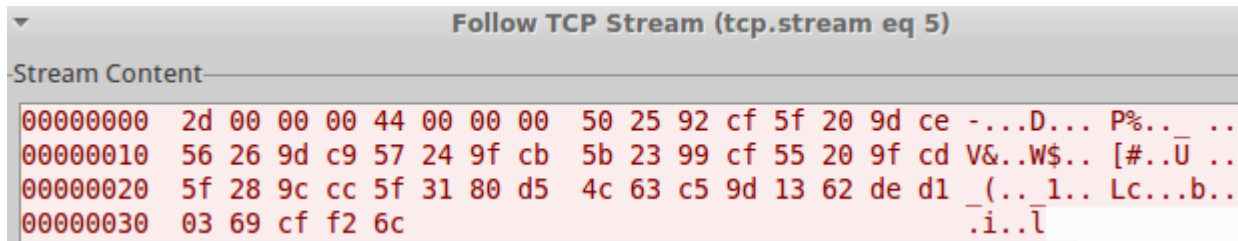


Figure 9: AbaddonPOS exfiltrating encoded credit card data to C2

The first four bytes of the network traffic are the length of the encoded data, while the following four bytes are the value of the process handle returned by OpenProcess. The subsequent bytes are the encoded exfiltrated data, which in a decoded state follows this format:

[credit card data] ***[process name]

To encode the data, the malware first XORs four bytes of the plaintext with the process handle, followed by a second XOR with a hardcoded 4-byte key. The exfiltration network traffic in Figure 9 is shown in its plaintext state in Figure 10.

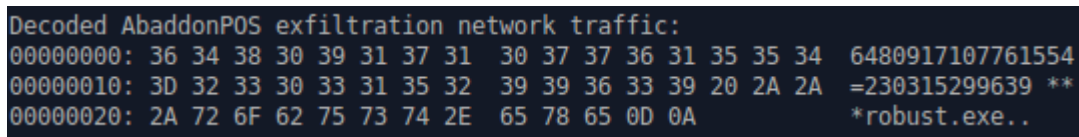


Figure 10: Plaintext exfiltrated credit card data and process name

The following Python script can be used to decode the network traffic, provided it has been encoded using the technique described above:

```
import sys,struct,hexdump

filename = sys.argv[1]

with open(filename, 'rb') as f:

    c2_traffic = f.read()

    encoded_size = struct.unpack('<I',c2_traffic[:4])[0]

    openprocess_handle = c2_traffic[4:8]

    encoded = c2_traffic[8:]

    key = [0x22,0x11,0xAA,0xFF]
```

```
decoded = ''

for i in range(encoded_size):

    decoded += chr((ord(encoded[i])^key[i%4])^ord(openprocess_handle[i%4]))

print 'Decoded AbaddonPOS exfiltration network traffic:'

hexdump.hexdump(decoded)
```

AbaddonPOS Variations

Of the samples Proofpoint researchers have discovered and analyzed so far, very few samples seem to have had any functionality added or removed. While “devil_host” is the most prominent mutex used by this malware, we have also found a sample that uses “devil_kor” (md5, a55843235cd8e36c7e254c5c05662a5b), and another that uses “DeviL_Task” (md5, ac03e0e9f70136adede78872e45f6182). We also observed a slightly updated version of AbaddonPOS (see *IOCs*) where almost all functionality was relocated to the encoded shellcode. In these updated samples the mutex “MG_REX” was used and the credit card search algorithm was also modified by adding ‘D’ as a valid track 2 delimiter.

Connecting the dots

TinyLoader has now been in development for at least a year, with a first sighting reported on January 16, 2015. Over the past year, TinyLoader has undergone several developmental changes, including:

- Switching from UDP protocol to TCP
- Removing process and UUID reporting
- Adding different anti-analysis
- Adding obfuscation and encoding

With the emergence of AbaddonPOS, it was quickly apparent that TinyLoader and AbaddonPOS are closely connected, and not simply because TinyLoader was used as the downloader. The code of TinyLoader and AbaddonPOS share some important similarities, including:

- Anti-analysis (CALL to push strings onto stack)
- Obfuscation (encoding shellcode using exact same encoding routine)

The similarities with code excerpts including a timeline according to Proofpoint data are provided below (Fig. 11).

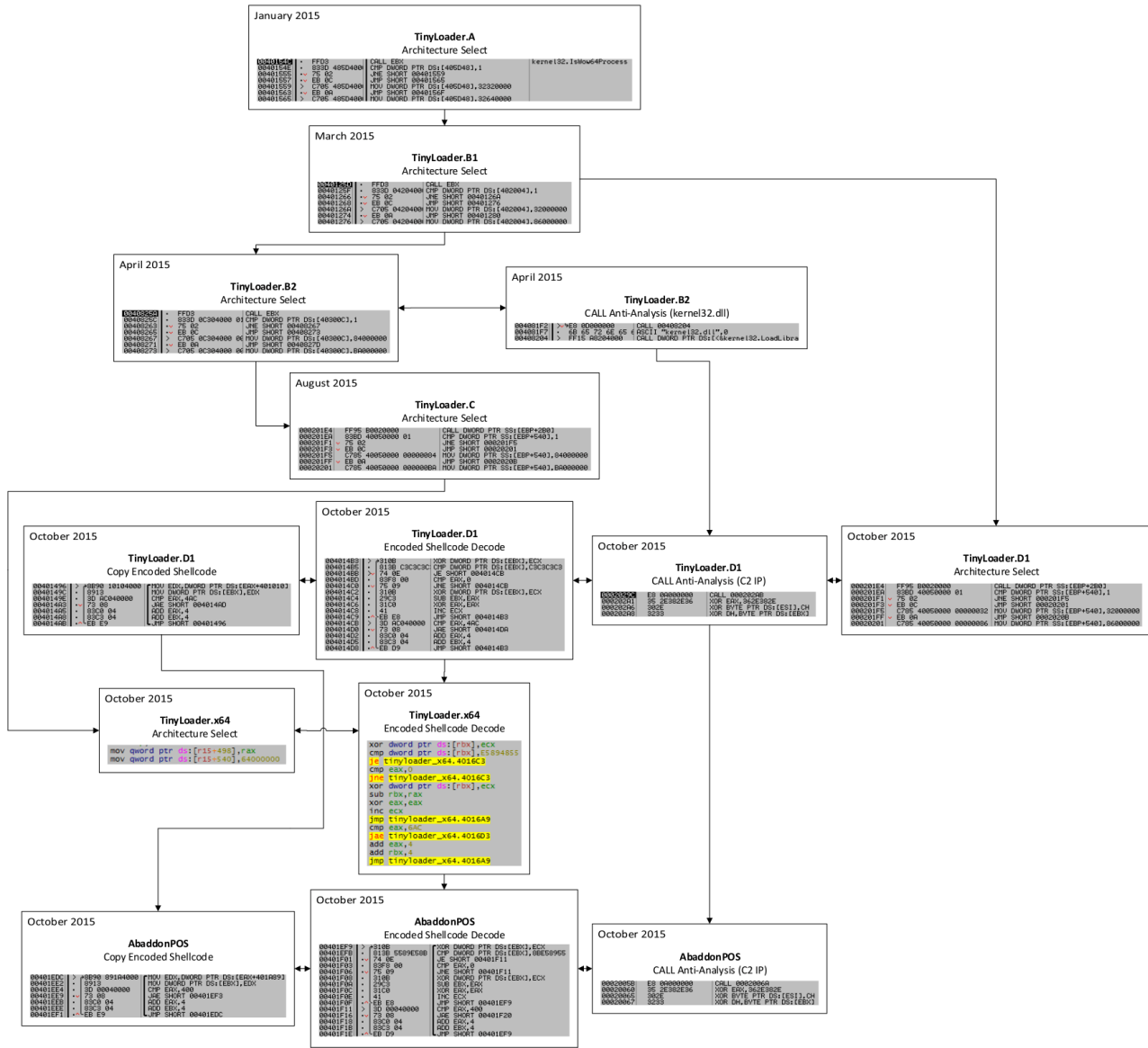


Figure 11: Code history comparison for TinyLoader and AbaddonPOS

Conclusion

The practice of threat actors to increase their target surfaces by leveraging a single campaign to deliver multiple payloads is by now a well-established practice. While using this technique to deliver point of sale malware is less common, the approach of the US holiday shopping season gives cybercriminals ample reason to maximize the return on their campaigns by distributing a new, powerful PoS malware that can capture the credit and debit card transactions of holiday shoppers.

UPDATE November 24, 2015

Further research on TinyLoader and AbaddonPOS turned up samples indicating that this threat has been in the wild since at least August 2015. The current earliest known samples of AbaddonPOS include:

266ce6d907a90e83da0083eee06af123 -> svchost_bin -> 50.7.138.138:13131 -> Compilation timestamp 2015-08-19 22:29:46

91992a1cac7f15e899b22d9a53cabf71 -> svchost_bin -> 50.7.124.172:13131

538482356b4eb4e0552d16b08d5c2908 -> svchost_bin -> 50.7.124.172:13131

05134cd6a50440b2c6d9ef62d2c2c3a3 -> svchost_bin -> 50.7.124.172:13131

7b137055fd40c39bdc76d27ff4fc82ed -> 50.7.124.172:15151 -> Location:

[hxxp://50.7.71[.]99/970/ad06b6e922623e436c7a.exe], downloaded by TinyLoader.C (md5: 4aa0ca129358b82a285e0d069a36e7fb)

7e49d646cb74718dcce21d3d3ad948d1 -> svchost_bin -> 50.7.124.172:14141 -> Location:

[hxxp://50.7.71[.]99/upload/7e49d646cb.exe], downloaded by TinyLoader.C (md5: 3733bb7a96e3091183d80b7a4914c830)

c7db01ba6b73188640e0fb65aab0d535 -> svchost_bin -> 50.7.124.172:15151

The earliest versions of AbaddonPOS are distinguished primarily by fact that it first targets track data delimiters ("=" and "^") for finding potential credit card data instead of a beginning number ("3", "4", "5", and "6").

Three earlier versions of AbaddonPOS have been identified (*credit: Nick Hoffman*):

81055d3e6ab2f349f334a87b090041dc -> svchost_bin -> 50.7.138[.]138:13030

da0cd8228745081b58594103163d22b8 -> svchost_bin -> 50.7.138[.]138:13030

04b68e4f4c7583201397d6674a3e2503 -> svchost_ghost -> 50.7.138[.]138:14040

The primary difference between these versions and the AbaddonPOS version analyzed in the original post is that these other versions contain a process blacklist: these processes will not be scanned for credit card data. The implementation is unique in that it searches only the first four bytes of each process; if those four bytes match, then it will search two more; and if those match as well, that process will be skipped. (Fig. 12) The blacklist contained the following partial process names:

svchso

iexplo

smss.e

csrss.

winlog

lsass.

spools

alg.ex

firefo

chrome
winini
steam.
skype.
dwm.ex.

```
0040172C | > 813D 3230400 | CMP DWORD PTR DS:[403032],68637673 | svch  
00401736 | .v 75 0F | JNE SHORT 00401747 |  
00401738 | . 66:813D 3630 | CMP WORD PTR DS:[403036],736F | so  
00401741 | .v 0F84 4701000 | JE 0040188E |
```

Figure 12: AbaddonPOS svchost.exe blacklist instructions

Proofpoint researchers discovered the following additional hashes for AbaddonPOS:

4a85feef07d4aed664624331cdbcdd66 -> DeviL_TasK -> 5.8.60[.]23:21920

6ac78bc0bd16273c654cec105567c73e -> no startup mutex -> 5.8.60[.]23:21930

6b02efef0580dce8e49d27196cff6825 -> M_RAY -> 193.28.179[.]13:20930

6f1d8ca36190668163f005c7f2c9007f -> M_RAY -> 193.28.179[.]13:20950

421dfc4856262445d12fe110bf4f2c56 -> DeviL_TasK -> 5.8.60[.]23:21940

9646e0a87be71c225f2aa8639354bd4f -> M_RAY -> 193.28.179[.]13:20940

46810f106dbaaff5c3c701c71aa16ee9 -> no startup mutex -> 176.114.0[.]165:21940

e9aeb88d393e6259b5fb520bc7a49ac0 -> M_REX -> 193.28.179[.]105:20910

Other malware that are likely used by these actor(s) include:

TinyLoader.C (md5: aa7897623f64576586e4b6ec99d8ccc6) was used to download Fleercivet/Bagsu, a Trojan used to commit adfraud (md5: 79dc1ce122f7bddd730d886df1a4739a, location: [hxxp://50.7.71[.]99/file/bin86crypt_full.exe])

TinyLoader.B (md5: a94c51c5e316d6e3b1cde1f80f99eb94) downloaded Fleercivet (md5: 637b764c78dda0e1d5351a10b19bcb8, location: [hxxp://50.7.71[.]214/upload/7777.exe])

TinyLoader.C (md5: 739cea68598ae347fae1d983e16a7d27) downloaded ReactorBot/Rovnix (md5: c755c9532c1ee517b25f98719968e154 and md5: 9a2fb9aa94d78313420c4106108b5fef, location: [hxxp://80.79.123[.]98/aurum/c.work.exe])

TinyLoader.C (md5: 19516ab9a7169c53bd811c975d5fea7d) was used to download Fleercivet (md5: 227e6b1f3e66f00a4fc683d4f39da904, location: [hxxp://50.7.143[.]61/id_1123.exe]) and a packed TinyLoader.C (md5: a86b91fda7ec634e44e4b6b7e69ed659, location: [hxxp://50.7.143[.]61/40930.exe])

These actors may have also employed CryptoWall at some point, as the imphash for 227e6b1f3e66f00a4fc683d4f39da904 matches the imphash for a known CryptoWall sample (md5: 2af149845f4d1ce8e712622d3f1ec46e). Both samples are packed, so it is possible that two actors utilized the same packer/crypter or packing/crypting service.

References

- [1] <https://www.washingtonpost.com/news/the-switch/wp/2014/08/22/secret-service-estimates-type-of-malware-that-led-to-target-breach-is-affecting-over-1000-u-s-businesses/>
- [2] <http://www.cio.com/article/2910024/data-breach/history-repeats-itself-as-pos-breaches-continue-in-2015.html>
- [3] <https://www.proofpoint.com/us/threat-insight/post/In-The-Shadows>
- [4] <http://www.cyphort.com/psychcentral-com-infected-with-angler-ek-installs-bedep-vawtrak-and-pos-malware/>
- [5] <http://researchcenter.paloaltonetworks.com/2015/10/understanding-and-preventing-point-of-sale-attacks/>

Indicators of Compromise (IOCs)

IDS/IPS Detection (ET signature IDs)

TinyLoader:

2020150-2020153,2020849-2020852,2812523,2812524,2814778,2814779,2814803

TinyDownloader (downloader shellcode HTTP request):

2814810

AbaddonPOS:

2814677-2814680

TinyLoader Samples:

0c77886a3ea42b75fcd860d4d97e72c5

a3ea1a008619687bdfef08d2af83f548

a53d8212a47bf25eeca87c1e27042686

a7a666ab9548fd1f0a8eb8050d8ca483

a9cc6736e573ad9e77359062e88114e2

aaac35389c9be79c67c4f5c4c630e5d5

b3a057f55a8fa2aad5b8d212a42b4a88

bcf271e83c964eb1fd89e6f1a7b7a62f

c42f20e2a68b8829b52b8399b7b33bf2
d785592932323f6ddaa121bcdcbceba0
e08aeb0bfcbae33b851af9f8be413111
e92254f9ce7d6f45e907e77de146ef37
ec322598eec364a755b5aea70d2a2da8
1c02f2f3fa15cc6a472119389d25983e
1c2a757c63ee418135e89cc8ef0d6e63
2b3704e0acbcabc265d0d08502a9bf373
3a7ac0c907b2c406ab480d4ed2f18161
3f71031ce8ecb0f48847ccb8be86a5fe
4b86cbb2e9f195bef3770d877206068d
6ee164908a94a881032d0649e2bd2505
6f7fabeb9ce76a1d52dbf5a40cbc74e8
7b7ffdd46d1f7ccea146fd9d5a2412ae
7c69dc17977b3431ff15c1ae5927ed0d
7eddbf17a3d1e398621194b0f22402a7
8d6d7a7d77215370d733bda57ef029f4
8df542e35225e0708cd2b3fe5e18ac79
9b340ac013c052ffb2beb29d26009a24
47e5c290f3f443cca027aa344cbf194f
54f1cda856ae921846e27f6d7cc3d795
77f124332a17b3ef6c0b6a799ad0c888
89a19ccb91977d8b1a020f580083d014
9320175f8af07503a2b2eb4d057bac07
885829081f91c6baf458166c3f42e281
a1d1ba04f3cb2cc6372b5986fadb1b9f

TinyLoader C2 IP addresses:

91.234.34[.]144

50.7.138[.]138

149.154.64[.]167

5.8.60[.]23

176.114.0[.]165

AbaddonPOS Samples:

5bf979f90307bac11d13be3031e4c6f9

a168fef5d5a3851383946814f15d96a7

a55843235cd8e36c7e254c5c05662a5b

1c19494385cb21b7e18252b5abd104f6

2b58f7cb4df18509a743226064b30675

752dcae6eb492263608a06489546098f

976275965fcf19a98da824b1959500c1

227e6b1f3e66f00a4fc683d4f39da904

8ca1278e2821fd2dd19c28725f754577

ac03e0e9f70136adede78872e45f6182

12cd4df2264624578919596371edee81

317f9c57f7983e2608d5b2f00db954ff

f63e0a7ca8349e02342c502157ec485d

0900582ba65c70a421b5d21d4ed21f16

4b0db5398f02dae5315f0baff1475807

703f492b2624899ec47b929f65265bbb

5e33b1273b2e2d4cd0986b9873ab4bc4

d11c4a4f76b2bea502b80229a83c30bc

e50edb61e796c6ead88cac53719e2d00

dc1a975e20eca705c6c78dc24f1290b5

6a6977ea317f0240a3dacc0753257518

5e06563f6303eab10c3cd46f0fd5c2d6

7ef654cdc7c2b54772400e26eb292caf

946be7ddd511ff9f49b5073896346eab

AbaddonPOS Exfiltration C2 IP addresses:

5.8.60[.]23:21910

5.8.60[.]23:21930

50.7.138[.]138:13030

50.7.138[.]138:15050

91.234.34[.]44:20940

91.234.34[.]44:20970

149.154.64[.]167:20910

149.154.64[.]167:20920

149.154.64[.]167:20940

149.154.64[.]167:20940

176.114.0[.]165:20910

176.114.0[.]165:21910

176.114.0[.]165:21940

Observed AbaddonPOS Location URLs:

[hxxp://50.7.143[.]61/f_p/f_940.exe]

[hxxp://50.7.143[.]61/n_p/n_940.exe]

[hxxp://50.7.143[.]61/kor_up.exe]

[hxxp://50.7.143[.]61/f_p/f_910.exe]

[hxxp://50.7.143[.]61/f_p/15050.exe]

[hxxp://50.7.143[.]61/a_p/a_970.exe]

[hxxp://50.7.143[.]61/x_file/x_910.exe]

[hxxp://50.7.143[.]61/x_file/x_930.exe]

[hxxp://50.7.143[.]61/files/p_910.exe]

[hxxp://50.7.143[.]61/a_p/a_970.exe]

[hxxp://50.7.138[.]138/file_x/x_910.exe]

[hxxp://50.7.138[.]138/file_x/x_930.exe]

[hxxp://50.7.138[.]138/n_940.exe]

[hxxp://50.7.138[.]138/n_910.exe]

[hxxp://50.7.71[.]99/explorer.exe]

AbaddonPOS Yara signature:

rule AbaddonPOS

{

meta:

description = "AbaddonPOS"

author = "Darien Huss, Proofpoint"

reference = "md5,317f9c57f7983e2608d5b2f00db954ff"

strings:

\$s1 = "devil_host" fullword ascii

\$s2 = "Chrome" fullword ascii

\$s3 = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run" fullword ascii

\$i1 = { 31 ?? 81 ?? 55 89 E5 8B 74 }

condition:

uint16(0) == 0x5a4d and (all of (\$s) or \$i1) and filesize <= 10KB*

}

Code Comparison Samples

TinyLoader.A,1e4906b4cfcad2e8d34a4937fa0c93e2

TinyLoader.B1,c0d530c9724d7c42adab3c7030a2383b

TinyLoader.B2,bd69714997e839618a7db82484819552

TinyLoader.C,739cea68598ae347fae1d983e16a7d27

TinyLoader.D1,7eddbf17a3d1e398621194b0f22402a7

TinyLoader.X64,b10444fcb83c03a5d6395831721fe750

AbaddonPOS,f63e0a7ca8349e02342c502157ec485d

Source: <https://www.proofpoint.com/us/threat-insight/post/AbaddonPOS-A-New-Point-Of-Sale-Threat-Linked-To-Vawtrak>