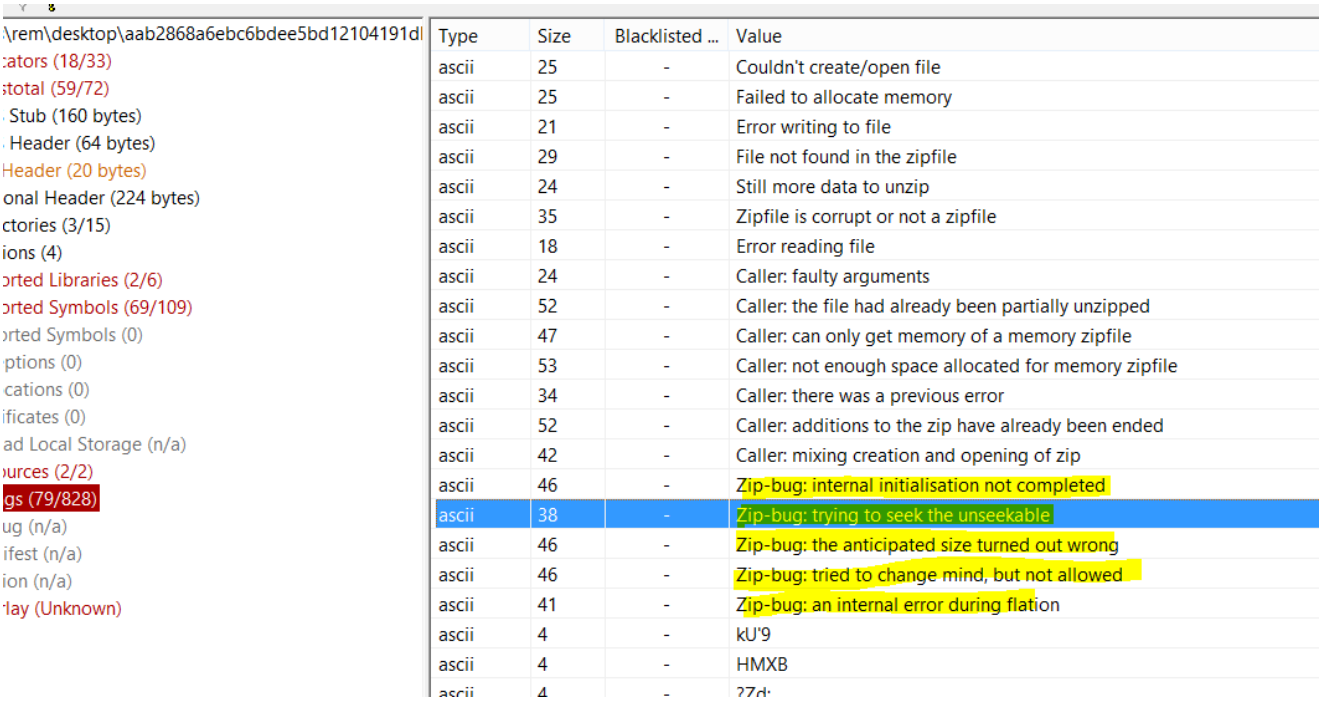
PebbleDash - Lazarus / HiddenCobra RAT

malwarenailed.blogspot.com/2020/06/peebledash-lazarus-hiddencobra-rat.html

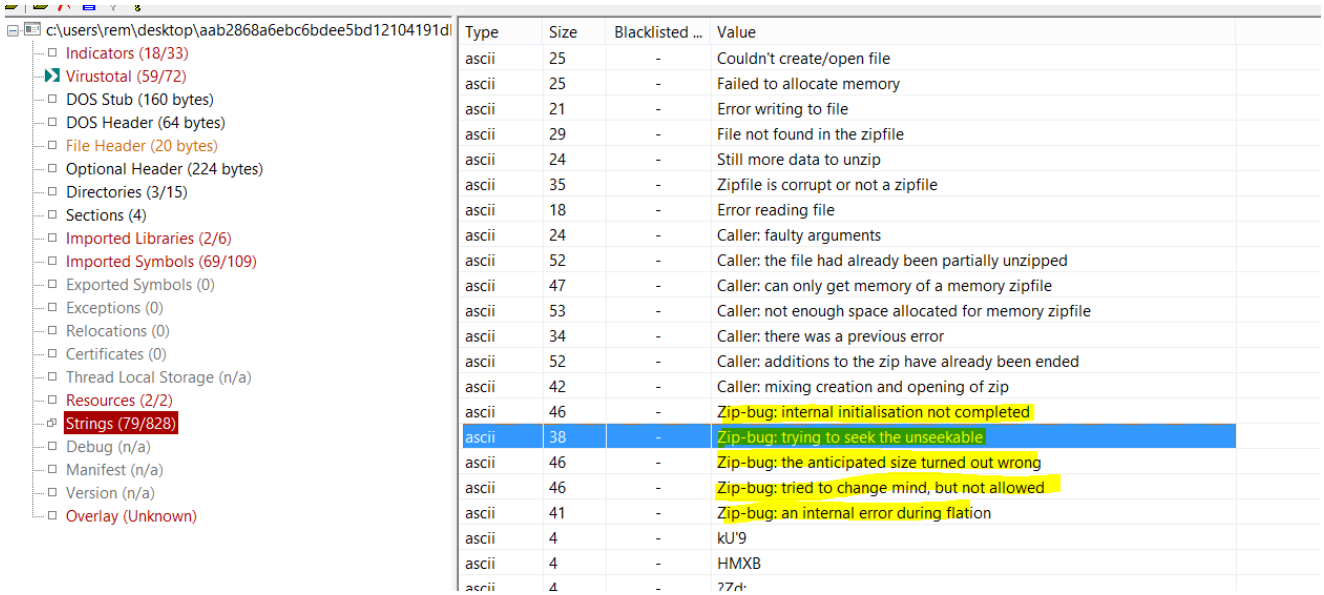
malwarenailed



Type	Size	Blacklisted ...	Value
ascii	25	-	Couldn't create/open file
ascii	25	-	Failed to allocate memory
ascii	21	-	Error writing to file
ascii	29	-	File not found in the zipfile
ascii	24	-	Still more data to unzip
ascii	35	-	Zipfile is corrupt or not a zipfile
ascii	18	-	Error reading file
ascii	24	-	Caller: faulty arguments
ascii	52	-	Caller: the file had already been partially unzipped
ascii	47	-	Caller: can only get memory of a memory zipfile
ascii	53	-	Caller: not enough space allocated for memory zipfile
ascii	34	-	Caller: there was a previous error
ascii	52	-	Caller: additions to the zip have already been ended
ascii	42	-	Caller: mixing creation and opening of zip
ascii	46	-	Zip-bug: internal initialisation not completed
ascii	38	-	Zip-bug: trying to seek the unseekable
ascii	46	-	Zip-bug: the anticipated size turned out wrong
ascii	46	-	Zip-bug: tried to change mind, but not allowed
ascii	41	-	Zip-bug: an internal error during flation
ascii	4	-	kU'9
ascii	4	-	HMXB

Hi folks. I was analyzing the PebbleDash malware used by Lazarus APT group. While analyzing the original sample (Md5: d2de01858417fa3b580b3a95857847d5), I was able to find out the C2 server and the port, where it intends to communicate to. I also found an interesting technique it uses to identify the OS version of the victim machine.

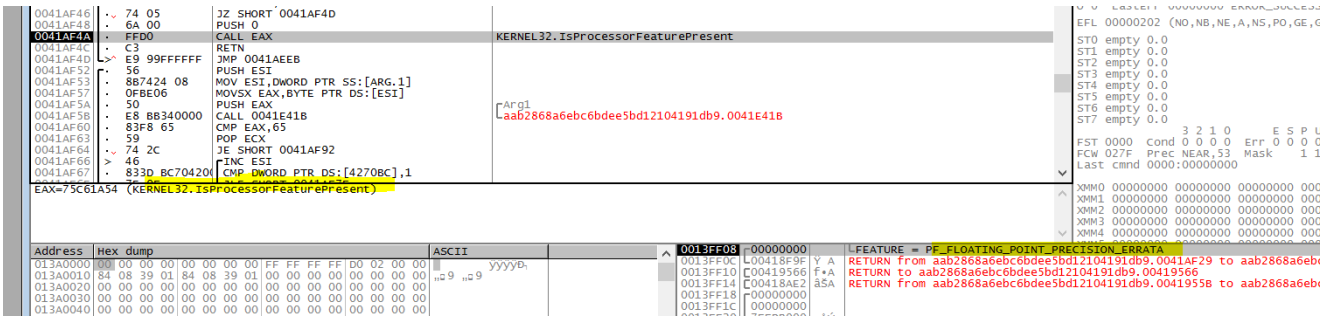
During static analysis, I observed interesting strings were starting with "Zip-bug", as can be seen below. Using yara rules I was able to discover some other samples uploaded to HA (Hybrid Analysis) with the same strings embedded. These samples seemed to be not related to d2de01858417fa3b580b3a95857847d5. However, they communicated to South Korea and China.



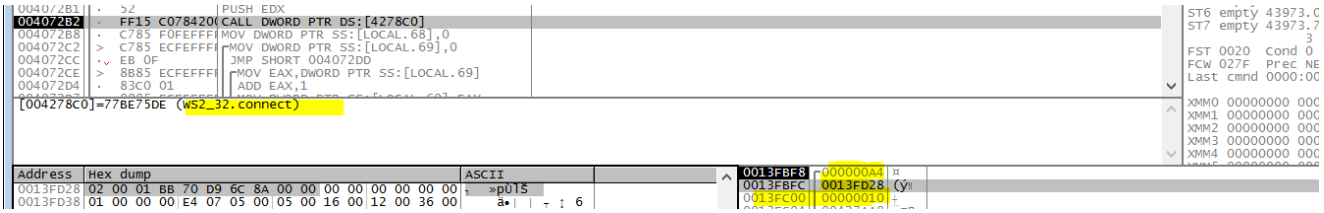
c:\users\rem\desktop\aab2868a6ebc6bdee5bd12104191dl
- Indicators (18/33)
- Virustotal (59/72)
- DOS Stub (160 bytes)
- DOS Header (64 bytes)
- File Header (20 bytes)
- Optional Header (224 bytes)
- Directories (3/15)
- Sections (4)
- Imported Libraries (2/6)
- Imported Symbols (69/109)
- Exported Symbols (0)
- Exceptions (0)
- Relocations (0)
- Certificates (0)
- Thread Local Storage (n/a)
- Resources (2/2)
- Strings (79/828)
- Debug (n/a)
- Manifest (n/a)
- Version (n/a)
- Overlay (Unknown)

Type	Size	Blacklisted ...	Value
ascii	25	-	Couldn't create/open file
ascii	25	-	Failed to allocate memory
ascii	21	-	Error writing to file
ascii	29	-	File not found in the zipfile
ascii	24	-	Still more data to unzip
ascii	35	-	Zipfile is corrupt or not a zipfile
ascii	18	-	Error reading file
ascii	24	-	Caller: faulty arguments
ascii	52	-	Caller: the file had already been partially unzipped
ascii	47	-	Caller: can only get memory of a memory zipfile
ascii	53	-	Caller: not enough space allocated for memory zipfile
ascii	34	-	Caller: there was a previous error
ascii	52	-	Caller: additions to the zip have already been ended
ascii	42	-	Caller: mixing creation and opening of zip
ascii	46	-	Zip-bug: internal initialisation not completed
ascii	38	-	Zip-bug: trying to seek the unseekable
ascii	46	-	Zip-bug: the anticipated size turned out wrong
ascii	46	-	Zip-bug: tried to change mind, but not allowed
ascii	41	-	Zip-bug: an internal error during flation
ascii	4	-	kU'9
ascii	4	-	HMXB
ascii	4	-	?7d·

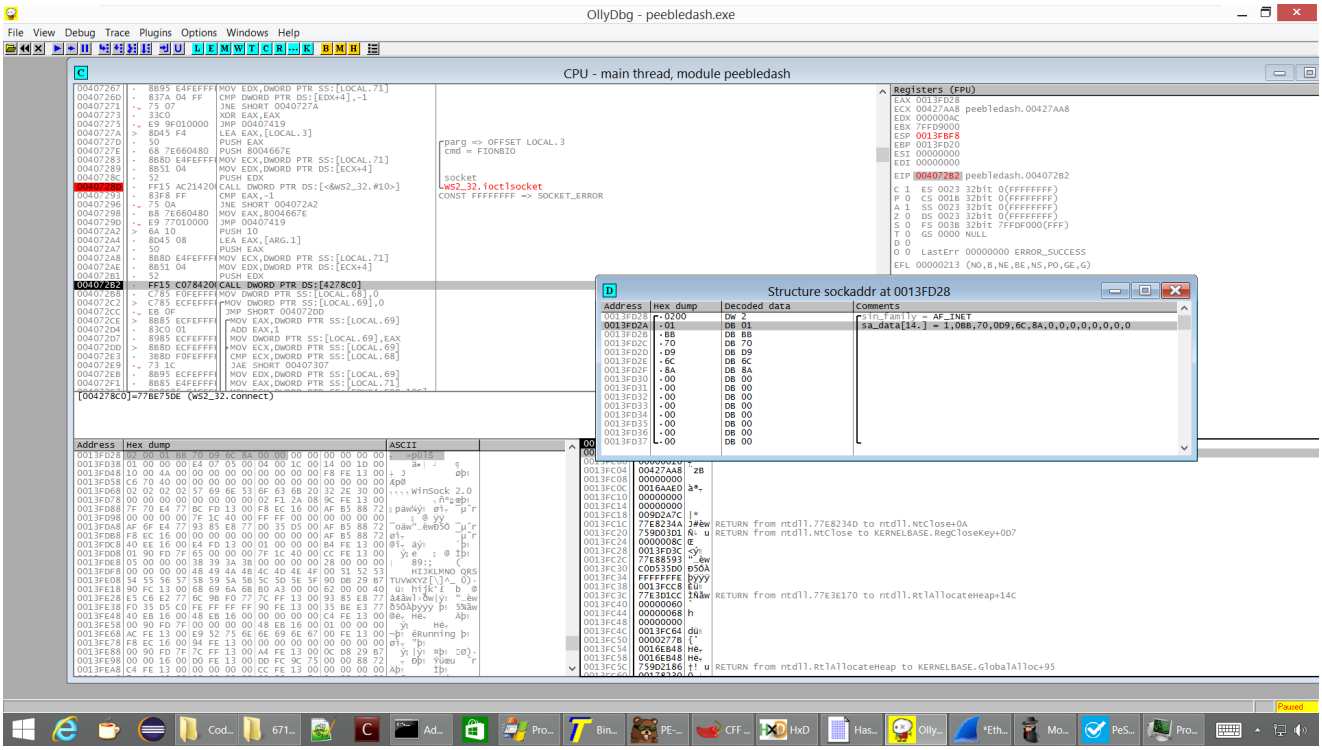
While performing dynamic anlaysis, I observed that the sample uses the API call IsProcessorFeaturePresent to determine the version of the victim OS. The PF_FLOATING_POINT_PRECISION_ERRATA feature is explicitly set to FALSE in x86 version 6.1 and higher.



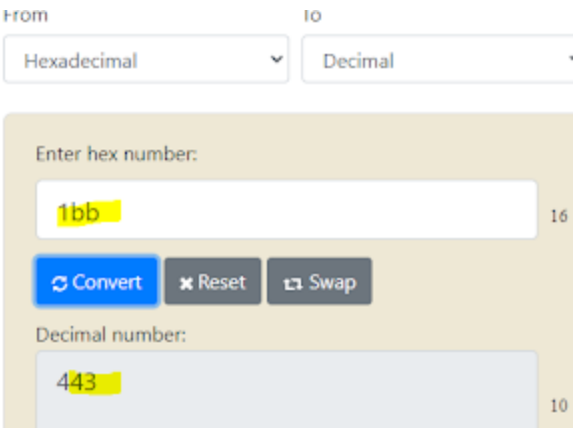
The sample loads several libraries dynamically during run time. This also included wsock32.dll. Malware usually does this as an anti-analysis technique (anti static analysis)



I decoded the "sockaddr" structure which is passed on to the connect function.



The first two bytes in the structure represents the destination port and we can see that it is 443 in this case.



The rest of the four bytes are: 0x70 0xd9 0x6C 0x8A, which translates to 112.217.108.138 (hex to decimal). This is the C2 ip address where PebbleDash communicates to. This IOC also be seen in the US-CERT advisroy.
https://www.us-cert.gov/ncas/analysis-reports/ar20-133c

terminate processes, and perform target system enumeration.

For a downloadable copy of IOCs, see MAR-10288834-3.v1.stix.

Submitted Files (1)

aab2868a6ebc6bdee5bd12104191db9fc1950b30bcf96eab99801624651e77b6 (D2DE01858417FA3B580B3A958

IPs (1)

112.217.108.138

Findings

aab2868a6ebc6bdee5bd12104191db9fc1950b30bcf96eab99801624651e77b6

Tags

rootkit trojan

Details

Name	D2DE01858417FA3B580B3A95857847D5
Size	167937 bytes

PebbleDash inserts fake "server name" in the TLS packet. We can see below some: