The InterPlanetary Storm: New Malware in Wild Using InterPlanetary File System's (IPFS) p2p network

by Anomali Threat Research

Summary

In May 2019, a new malware was found in the wild that uses a peer-to-peer (p2p) network on top of InterPlanetary File System's (IPFS) p2p network. The malware found in the wild targets Windows machines and allows the threat actor to execute any arbitrary PowerShell code on the infected machines. The use of a legitimate p2p network can make it difficult to discover the malicious traffic as it potentially is blended in with legitimate traffic to the legitimate p2p network. It can also make it harder to sinkhole the botnet since there is a risk the legitimate p2p network is also taken down with it.

Introduction

When a threat actor wants to commandeer a machine, a Command and Control (C2) communication channel needs to be established. With this communication channel, the threat actor can send commands to the infected machines and the response can be sent back from the infected machines to the actor. In general, there are two types of schemes that can be used. The first type is a p2p model, and the second type is a server-client model.

Server-Client Model

In a server-client model, the infected machines connect to a set of C2 servers that provides the infected machines with instructions and handles the responses sent back. A threat actor may use these C2 servers directly to control the infected machines by having the administrator panel hosted on these servers. Another option is that the threat actor uses a second layer of C2 servers. These servers never talk directly to the infected machines. Instead, the commands are sent from the second layer to the first, which relays the commands and the responses between infected machines and the second row of C2 servers, acting more like a proxy. This has the potential of protecting the second layer from being detected and possibly taken over by the authorities. Using two layers of C2 servers may also increase stealth. The threat actor can, for example, use a non-malicious website to proxy or relay the communication.

In recent years, threat actors have started using legitimate web services for C2 communications. For example, Twitter, Github, and Instagram are legitimate services that have been misused by threat actors. [1] The use of legitimate services have multiple advantages to the threat actor. By default, all of these services are using encrypted communication via TLS. This means the threat actors do not need to configure and maintain certificates because it is all done for them. Another benefit is that these services are common on home and corporate networks. The malware talks to the same servers as normal web browsers visiting the websites, essentially allowing the malicious traffic to blend in with the legitimate

communications. While this method allows malware traffic to hide better, it is easier for authorities to take down. One way of making takedowns harder is to use a p2p model instead of a server-client model.

P2P Model

In a p2p model, the infected machines are not necessarily communicating directly with servers controlled by the threat actor. Instead, the machines are connected directly to each other via a p2p network, commonly called a p2p botnet. All the threat actor has to do is send a single command to one infected machine and that command will automatically propagate to all other infected machines. A p2p model is generally more difficult for a threat actor to implement because there are different problems that need to be solved than when using the server-client model. The first problem is bootstrapping. How does the newly infected machine find other infected machines to connect to? One method is to include a list of known peers in the malware that it can connect to. On the defender side, a known bootstrapping list can be used for both detection and prevention; blocking access to the machines on the list at the perimeter. The other problem the threat actor needs to solve is Network Address Translation (NAT) traversal for machines not directly connected to the internet.[2] A p2p network works by having the peers connected to each other. If a machine is not directly connected to the internet but instead is connected via a NAT router, it cannot be reached by a machine on the internet. This is because the NAT router will only let through communication that is initiated from behind the NAT. This means if the p2p network has all nodes behind NAT, none of the nodes would be able to connect to each other.

When comparing to the server-client model, p2p botnets do not have the opportunity to hide their traffic amongst legitimate traffic. Up until recently, p2p communication on a corporate network could be taken as suspicious activity. In present day, more and more legitimate services are utilizing p2p technology that is slowly creeping into the enterprise space. For example, Microsoft Windows 10 has a feature called “Delivery Optimization” that delivers updates to machines via a p2p network established by machines connected to the same Active Directory domain.[3] Similar to misusing web services to hide malicious traffic, threat actors misuse legitimate p2p network to hide their traffic. In addition to blending with the normal traffic, the botnet is intertwined with the legitimate nodes in such a way making it impossible to take down the malicious botnet without taking down the legitimate p2p network. In May 2019, a new botnet was discovered that utilizes the IPFS p2p network.

IPFS is a project that aims to improve today’s internet by making it more decentralized.[4] The project is designed to be a distributed p2p filesystem, and the filesystem can be used to host any kind of files, including static web pages that can be viewed with a web browser. The files hosted on IPFS can be accessed by using a client or via public gateways. For example, Cloudflare runs a public IPFS gateway. [5] The network code for IPFS is released as an open source project called “libp2p”, which is a modular network stack that allows anyone to take advantage of the network code used by IPFS (7). The library’s support includes bootstrapping, NAT-traversal, relays, peer discovery, pubsub functionality. It can be used to construct an independent p2p network by providing bootstrapping nodes.The library also includes IPFS’s bootstrapping nodes that can be used to layer the new p2p network on top of IPFS’s p2p network. A functionality that can be appealing to threat actors.

The malware discovered in May 2019 by Anomali Threat Research, does use libp2p to layer its p2p network on top of IPFS’s. The malware has been named IPStorm (InterPlanetary Storm) from its use of IPFS’s p2p network and the project name used by the threat actor.

Technical breakdown

IPStorm is a malware written in Go (Golang). The samples found in the wild have been targeting the Windows operating system. The analyzed binaries include the path “/Users/brokleg/go/src/storm/” which suggests it has been developed on a macOS machine and the malware author has named the project “storm”, possibly after 2007’s worm named Storm that used a p2p network for C2 communication. The malware is a large, with the unpacked binary being around 15 MB in size. The code is split up into multiple Go packages. The packages are listed below:

- storm/avbypass
- storm/backshell
- storm/ddb
- storm/filetransfer
- storm/logging
- storm/node
- storm/powershell
- storm/util
- storm/ddbinterface
- storm/nodeinterface

The malware has some simple antivirus (AV) evasion techniques. It uses sleeps, memory allocations and generation of random numbers. The “allocateMemory” function is very simple. The core function body is shown in Figure 1. It allocates 100 byte arrays with a size of 3 MB each.

```
   ,==< 0x00742190      763e            jbe 0x7421d0                ;[1]
   |:   0x00742192      83ec14          sub esp, 0x14
   |:   0x00742195      31c0            xor eax, eax
  ,===< 0x00742197      eb2e            jmp 0x7421c7                ;[2]
.----> 0x00742199      89442410        mov dword [var_10h], eax
:||:   0x0074219d      8d05200e9900    lea eax, sym.type.uint8     ; 0x990e20
:||:   0x007421a3      890424          mov dword [esp], eax
:||:   0x007421a6      c7442404c0c6.   mov dword [var_4h], 0x2dc6c0    ; [0x2dc6c0:4]=-1
:||:   0x007421ae      c7442408c0c6.   mov dword [var_8h], 0x2dc6c0    ; [0x2dc6c0:4]=-1
:||:   0x007421b6      e86579cfff      call sym.runtime.makeslice  ;[3]
:||:   0x007421bb      8b44240c        mov eax, dword [var_ch]     ; [0xc:4]=-1 ; 12
:||:   0x007421bf      c60001          mov byte [eax], 1
:||:   0x007421c2      8b442410        mov eax, dword [var_10h]    ; [0x10:4]=-1 ; 16
:||:   0x007421c6      40              inc eax
:||:   ; CODE XREF from sym.storm_avbypass.allocateMemory (0x742197)
:`---> 0x007421c7      83f864          cmp eax, 0x64               ; 'd' ; 100
`====< 0x007421ca      7ccd            jl 0x742199                 ;[4]
   |:   0x007421cc      83c414          add esp, 0x14
   |:   0x007421cf      c3              ret
```

Figure 1: Showing memoryAllocation function. The function creates 100 (0x64) byte arrays with space for 3 MB (0x2dc6c0).

Instead of using Mutexes or window names to ensure singe execution, IPStorm uses the third-party package “single” (github.com/marcsauter/single). Single uses lock files to ensure only one instance is running. The “single” name used by the malware is “n3R1PYfY”, the lock file is placed in the %TMP% folder (%TMP% 3R1PYfY.lock). When the malware is sure only one instance is running, it performs an enumeration of the infected machine. It uses the third-party package goInfo

(github.com/matishsiao/goInfo) and PowerShell commands to collect most of the information. The collected user information is published to the p2p network is shown in the struct below:

```
type node.NodeInfo struct {
    HostID string
    Version string
    Platform string
    SystemInfo goInfo.GoInfoObject
    Uid string
    Gid string
    UserName string
    UserDisplayName string
    UserHomeDir string
    IsAdmin bool
    ExecutablePath string
    ComputerID string
}
```

To ensure the malware can connect to the p2p network, it adds a rule to the firewall. For the networking part, the malware uses “libp2p”. The underlying protocol used by the library is “protobuf”. The malware uses the PubSub functionality provided by the project. It uses two topics: “info” and “cmd”. To find other peers, it uses libp2p’s support for distributed hash tables (DHT). The new bot uses a hardcoded string to advertise its presence and to find other peers.

The malware has support for downloading and uploading files. It is performed by sending the content over the PubSub network. Each bot in the network serve its executable file and the threat actor uses this method to distribute newer versions of the bot. It also has a “reverse shell” (called “backshell” by the author) functionality. With this functionality, the threat actor can execute any arbitrary PowerShell code on the infected machine. The malware installs itself under the following location:

```
\.PHYSICALDRIVE0AppDataLocalPackages%s_%sAppData
```

For the first “%s”, the malware does a random selection from one of the folder names in the list below.

- Microsoft.AAD.BrokerPlugin
- Microsoft.AccountsControl
- Microsoft.AsyncTextService
- Microsoft.BioEnrollment
- Microsoft.CredDialogHost
- Microsoft.ECApp
- Microsoft.LockApp
- Adobe.Photoshop
- Adobe.Illustrator
- Adobe.Reader

The second “%s” is replaced with a random string of 13 characters. The malware selects characters from the following alphabet:
“qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890”.

The name of the executable is randomly selected from the following list:

- svchost
- csrss
- rundll32
- winlogon
- smss
- taskhost
- unsecapp
- AdobeARM
- winsys
- jusched
- BCU
- wscntfy
- conhost
- csrss
- dwm
- sidebar
- ADService
- AppServices
- acrotray
- ctfmon
- lsass
- realsched
- spoolsv
- RTHDCPL
- RTDCPL
- MSASCui

For persistence, the malware adds an entry to “HKCU:SoftwareMicrosoftWindowsCurrentVersionRun”.

Analysis

The malware has a relatively simple set of core functionalities but utilizes a complex network stack. The use of IPFS’s network stack allows the malicious botnet to hide within a legitimate p2p network. This makes it unclear which peers are infected bots or legitimate IPFS peers. Just as an example during the bootstrap processing, the malware uses the same bootstrapping peers as the IPFS network does. It is likely the bots are in the development process and more functions will be added in the future. Bot version found in the analyzed samples ranges from “0.0.2n” to “0.0.2y”, suggesting early stage of the bot evolution.

From the analysis of an estimated source code tree structure, generated using metadata in the samples, it is possible the malware either can be compiled for other operating systems than Windows or are in the process of being developed for other operating systems. The tree structure below shows that most of the code for the “backshell” was located in the file “backshell.go” while one function is located in the file named “backshell_windows.go”. This suggests the author uses Go’s build tags support to select which

“openLocalShell” function to compile depending on the target operating system. It can also be concluded that the malware was compiled on a macOS machine.

```
Package storm/backshell: /Users/brokleg/go/src/storm/backshell
File: backshell.go
       (init)ializers Lines: 16 to 16 (0)
       StartServer Lines: 18 to 23 (5)
       handleStream Lines: 23 to 36 (13)
       handleLocalShellIn Lines: 36 to 63 (27)
       handleLocalShellOut Lines: 63 to 64 (1)
File: backshell_windows.go
       openLocalShell Lines: 11 to 16 (5)
```

The botnet size is likely in the lower thousands because during a ten-hour collection, 2847 unique p2p nodes announced themselves with the identifier used by the malware and were added to the DHT. This number does not correspond the number of bots in the botnet since the bot generates a new libp2p id each time it is started up but is a good estimate of the upper bound of the botnet size. The size of the botnet may be indicative of it still being in its early stage of evolution.

Conclusion

In general, the network architecture for botnets can be described either as a server-client model or a p2p model. The p2p model is usually much harder to implement but also more resilient against take-downs. A downside to the p2p model is it’s noisier than the server-client model. The server-client model uses legitimate web services to blend hide its traffic among the legitimate traffic. In May 2019, a new malware was discovered that uses IPFS’s p2p network to hide its p2p traffic. This is the first malware found in the wild that is using IPFS’ p2p network for its C2 communication. By using a legitimate p2p network, the malware can hide its network traffic among legitimate p2p network traffic. This method also provides some protection against takedowns, since sinkholing the p2p network potentially could take down the whole IPFS network. With a good open source library that allows any threat actor to implement a p2p network with relative ease, is this just the beginning of a new evolution of p2p botnets?

Threatstream enterprise users can read a more detailed analysis here.

IOCs

- 2545175418021b0bcdce1fa055dc292500fa9895857c6df86461f9d74a342d15
- 41ec6577b3a362cf9e5b136ca3971204147bc6c171b65bab3546f631a5c2efe0
- 49c3fa3a2b7b5894559a28456ad611fa4692f72c1ec86eee925df81735278d53
- 5918446d82bd8d6c40c6cdf5ab70a012f27939e9154e26f79a7c9870811cad8c
- 5d980fa37aef47022941f6afffe718ed0eed4de746edd7f494a407f84e75fd50
- 7f731d2502dd39cbc16193ca7e9d147fe158c10236e00c634bb0680e2bfc4bfa
- 8531921258132a4eb9b4b4545e85c72c2815d53c22f72f92822c199a15562a7c
- 8b24e640338654fbda233f50766136a9cf33f1a5444fd98162edbc2bdaf324c2
- c0cfc62c5c349523884d502088234026c3eb67c802d7d02018e7aa9337930e0b
- c19b34621a7a57d831113e0b854bbc1fa7217d578ca9bd477d805f56a73a0100
- cc124869cbc871ec99d3470f2287df816a5f75a05eaf79df65a8ecf9a8283eb9

- f987928992a9c43c65429d2abe93b4683a027b520d79246248fca49c31f234d7
- fe8dc4955f2e04bbf800e913d29848c2b99999554a802226e72ce68b6d8c71f7
- 01658e8cc706056aca50304b29bfcab92a60cd38771f2b3069ba94f525661122
- 039ae75eff58a5b1125c6d3248dfce338573a9040709b3a9e8cf910a1b4ada75
- 2accb615e375b99bb92c67655ad9558c44782148d8264ef94e5a0a7d4dc1c5da
- 2e751a9874267b100b10196ebbef58cfcf46644651b099c8a9cd1dd7cee64e8f
- 3276bcd9b95a28234c1ea5205d8c2358f5040e694331a16cef09d273b6238178
- 365480a61e39b67234692ba8b75b95749434aba28c952ac32f70ed777d25810a
- 36d652460b71f6e9aca16ba331197fc384f323f70ea879ec8ea40a7a5f554a7e
- 4549a271e2c4b157c8abe099d2e84c8f3d7c18782956267429e0be6e17eaed41
- 500f004eb51f282c14939ba52a5d85ac71f85e1f0ede00803a6ae3a2f01c272d
- 66d3799fc8132376d87f326f0131c41afecef560b0368bc54cc27b816025d9e6
- 948042a3071a4a6e1063f2e09717fd70f23e2c37cbdf024ccded76477b94778e
- a2d36f477e2a87acab32a1988f77b00ce9f24d61a4a21e63a8c67747ff2633fd
- af32455f41865094da020cad2675775bddd907020ed12d4ed68de5051ed62643
- de0b86fe66be7ef3f30f28d514c8bbced2973e78d9473671f16f4365e05e5a99
- e093152a3e44279259839e1c35eeaeb5ea0803611747b8b38acaffd65a83f83a
- f5ac2e7a6cfe6ce576a9c0df3721f90c5be93b5e4694716470ec7ea3c7492f59

Mitre ATT&CK

- T1059
- T1132
- T1041
- T1112
- T1086
- T1012
- T1060
- T1045

References

1. Jean-Ian Boutin, “Turla’s watering hole campaign: An updated Firefox extension abusing Instagram,” ESET, accessed May 28, 2019, published June 6, 2017, https://www.welivesecurity.com/2017/06/06/turlas-watering-hole-campaign-updated-firefox-extension-abusing-instagram/; FireEye Threat Intelligence, “HAMMERTOSS: Stealthy Tactics Define a Russian Cyber Threat Group,” FireEye Blog, accessed May 28, 2019, published July 29, 2015, https://www.fireeye.com/blog/threat-research/2015/07/hammertoss_stealthy.html.
2. James Wyke, “The ZeroAccess Botnet – Mining and Fraud for Massive Financial Gain,” Sophos, accessed May 28, 2019, published September 2012, https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/Sophos_ZeroAccess_Botnet.pdf.
3. Windows IT Pro Center, "Delivery Optimization for Windows 10 updates," Microsoft, accessed May 29, 2019, published May 31, 2019, https://docs.microsoft.com/en-us/windows/deployment/update/waas-delivery-optimization.
4. “A modular network stack,” Protocol Labs, accessed May 31, 2019, https://ipfs.io.

5. “Distributed Web Gateway,” Cloudflare, accessed publication May 31, 2019, https://www.cloudflare.com/distributed-web-gateway.
6. “A modular network stack,” Protocol Labs, accessed May 31, 2019, https://ipfs.io.