

NETMANAGEIT

Intelligence Report

Smargft Harnesses EtherHiding for Stealthy C2 Hosting

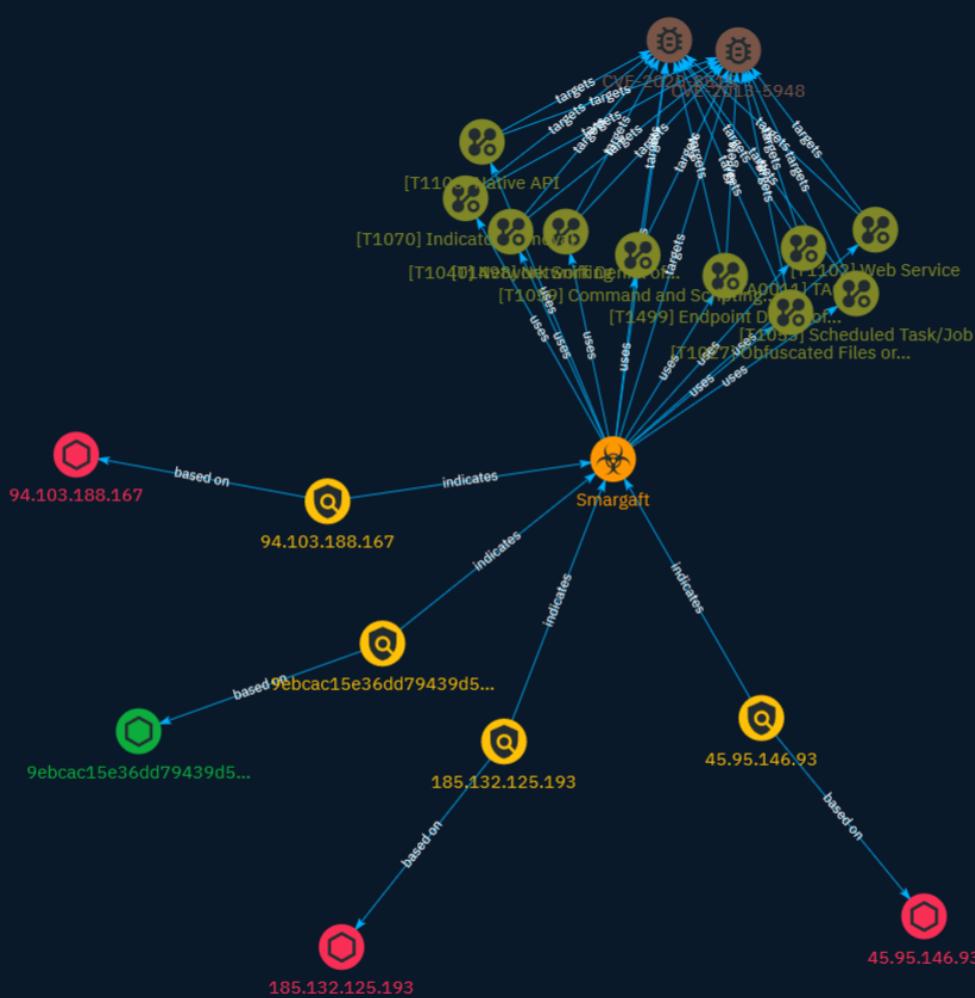


Table of contents

Overview

● Description	4
● Confidence	4
● Content	5

Entities

● Indicator	6
● Vulnerability	11
● Malware	12
● Attack-Pattern	13

Observables

● StixFile	29
● IPv4-Addr	30

External References

- External References

31

Overview

Description

Smargaft uses the Binance Smart Chain to host commands and control(C2) server, and it spreads through Shell scripts to keep itself going. Because of its smart use of contracts and Gafgyt's methods, we've decided to call it Smargaft. It mainly does DDoS attacks, runs system commands, and lets users connect anonymously using socks5 proxy.

Confidence

This value represents the confidence in the correctness of the data contained within this report.

15 / 100

Content

N/A

Indicator

Name
9ebcac15e36dd79439d5f92feb1933e0a68e75cc1e9eecb6369e4b3317e2949
Description
SUSP_ELF_LNX_UPX_Compressed_File SHA256 of 7f741495f14c828c20db4de6251673fd
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '9ebcac15e36dd79439d5f92feb1933e0a68e75cc1e9eecb6369e4b3317e2949']
Name
94.103.188.167
Description
- **Zip Code:** N/A - **ISP:** Alexhost - **ASN:** 200019 - **Organization:** Alexhost - **Is Crawler:** False - **Timezone:** Europe/Moscow - **Mobile:** False - **Host:** node-2-md - **Proxy:** True - **VPN:** True - **TOR:** False - **Active VPN:** False - **Active TOR:** False - **Recent Abuse:** False - **Bot Status:** False - **Connection Type:** Premium

required. - **Abuse Velocity:** Premium required. - **Country Code:** RU - **Region:** Moscow - **City:** Moscow - **Latitude:** 55.75 - **Longitude:** 37.62

Pattern Type

stix

Pattern

```
[ipv4-addr:value = '94.103.188.167']
```

Name

45.95.146.93

Description

- **Zip Code:** N/A - **ISP:** Alsycon - **ASN:** 49870 - **Organization:** Alsycon - **Is Crawler:** False - **Timezone:** Europe/Amsterdam - **Mobile:** False - **Host:** hosted-by.alsycon.net - **Proxy:** True - **VPN:** True - **TOR:** False - **Active VPN:** False - **Active TOR:** False - **Recent Abuse:** False - **Bot Status:** False - **Connection Type:** Premium required. - **Abuse Velocity:** Premium required. - **Country Code:** NL - **Region:** North Holland - **City:** Amsterdam - **Latitude:** 52.37 - **Longitude:** 4.89

Pattern Type

stix

Pattern

```
[ipv4-addr:value = '45.95.146.93']
```

Name

185.132.125.193

Description

- **Zip Code:** N/A - **ISP:** M247 Europe - **ASN:** 9009 - **Organization:** M247 Europe
 - **Is Crawler:** False - **Timezone:** Asia/Hong_Kong - **Mobile:** False - **Host:** 185.132.125.193 - **Proxy:** True - **VPN:** True - **TOR:** False - **Active VPN:** True - **Active TOR:** False - **Recent Abuse:** False - **Bot Status:** False - **Connection Type:** Premium required. - **Abuse Velocity:** Premium required. - **Country Code:** HK
 - **Region:** Central and Western District - **City:** Hong Kong - **Latitude:** 22.28 - **Longitude:** 114.18

Pattern Type

stix

Pattern

[ipv4-addr:value = '185.132.125.193']

Name

9ebcac15e36dd79439d5f92feb1933e0a68e75cc1e9eecb6369e4b3317e2949

Description

SUSP_ELF_LNX_UPX_Compressed_File SHA256 of 7f741495f14c828c20db4de6251673fd

Pattern Type

stix

Pattern

[file:hashes!'SHA-256' = '9ebcac15e36dd79439d5f92feb1933e0a68e75cc1e9eecb6369e4b3317e2949']

Name

94.103.188.167

Description

- **Zip Code:** N/A - **ISP:** Alexhost - **ASN:** 200019 - **Organization:** Alexhost - **Is Crawler:** False - **Timezone:** Europe/Moscow - **Mobile:** False - **Host:** node-2-md - **Proxy:** True - **VPN:** True - **TOR:** False - **Active VPN:** False - **Active TOR:** False - **Recent Abuse:** False - **Bot Status:** False - **Connection Type:** Premium required. - **Abuse Velocity:** Premium required. - **Country Code:** RU - **Region:** Moscow - **City:** Moscow - **Latitude:** 55.75 - **Longitude:** 37.62

Pattern Type

stix

Pattern

```
[ipv4-addr:value = '94.103.188.167']
```

Name

45.95.146.93

Description

- **Zip Code:** N/A - **ISP:** Alsycon - **ASN:** 49870 - **Organization:** Alsycon - **Is Crawler:** False - **Timezone:** Europe/Amsterdam - **Mobile:** False - **Host:** hosted-by.alsycon.net - **Proxy:** True - **VPN:** True - **TOR:** False - **Active VPN:** False - **Active TOR:** False - **Recent Abuse:** False - **Bot Status:** False - **Connection Type:** Premium required. - **Abuse Velocity:** Premium required. - **Country Code:** NL - **Region:** North Holland - **City:** Amsterdam - **Latitude:** 52.37 - **Longitude:** 4.89

Pattern Type

stix

Pattern

```
[ipv4-addr:value = '45.95.146.93']
```

Name

185.132.125.193

Description

- **Zip Code:** N/A - **ISP:** M247 Europe - **ASN:** 9009 - **Organization:** M247 Europe
- **Is Crawler:** False - **Timezone:** Asia/Hong_Kong - **Mobile:** False - **Host:**
185.132.125.193 - **Proxy:** True - **VPN:** True - **TOR:** False - **Active VPN:** True -
Active TOR: False - **Recent Abuse:** False - **Bot Status:** False - **Connection**
Type: Premium required. - **Abuse Velocity:** Premium required. - **Country Code:** HK
- **Region:** Central and Western District - **City:** Hong Kong - **Latitude:** 22.28 -
Longitude: 114.18

Pattern Type

stix

Pattern

```
[ipv4-addr:value = '185.132.125.193']
```

Vulnerability

Name
CVE-2013-5948
Name
CVE-2020-8515
Description
DrayTek Vigor3900, Vigor2960, and Vigor300B routers contain an unspecified vulnerability that allows for remote code execution.
Name
CVE-2013-5948
Name
CVE-2020-8515
Description
DrayTek Vigor3900, Vigor2960, and Vigor300B routers contain an unspecified vulnerability that allows for remote code execution.

Malware

Name

Smargaft

Name

Smargaft

Attack-Pattern

Name
TA0011
ID
TA0011
Name
Network Denial of Service
ID
T1498
Description
Adversaries may perform Network Denial of Service (DoS) attacks to degrade or block the availability of targeted resources to users. Network DoS can be performed by exhausting the network bandwidth services rely on. Example resources include specific websites, email services, DNS, and web-based applications. Adversaries have been observed conducting network DoS attacks for political purposes(Citation: FireEye OpPoisonedHandover February 2016) and to support other malicious activities, including distraction(Citation: FSISAC FraudNetDoS September 2012), hacktivism, and extortion. (Citation: Symantec DDoS October 2014) A Network DoS will occur when the bandwidth capacity of the network connection to a system is exhausted due to the volume of malicious traffic directed at the resource or the network connections and network devices the resource relies on. For example, an adversary may send 10Gbps of traffic to a server

that is hosted by a network with a 1Gbps connection to the internet. This traffic can be generated by a single system or multiple systems spread across the internet, which is commonly referred to as a distributed DoS (DDoS). To perform Network DoS attacks several aspects apply to multiple methods, including IP address spoofing, and botnets.

Adversaries may use the original IP address of an attacking system, or spoof the source IP address to make the attack traffic more difficult to trace back to the attacking system or to enable reflection. This can increase the difficulty defenders have in defending against the attack by reducing or eliminating the effectiveness of filtering by the source address on network defense devices. For DoS attacks targeting the hosting system directly, see [Endpoint Denial of Service](<https://attack.mitre.org/techniques/T1499>).

Name

Endpoint Denial of Service

ID

T1499

Description

Adversaries may perform Endpoint Denial of Service (DoS) attacks to degrade or block the availability of services to users. Endpoint DoS can be performed by exhausting the system resources those services are hosted on or exploiting the system to cause a persistent crash condition. Example services include websites, email services, DNS, and web-based applications. Adversaries have been observed conducting DoS attacks for political purposes(Citation: FireEye OpPoisonedHandover February 2016) and to support other malicious activities, including distraction(Citation: FSISAC FraudNetDoS September 2012), hacktivism, and extortion.(Citation: Symantec DDoS October 2014) An Endpoint DoS denies the availability of a service without saturating the network used to provide access to the service. Adversaries can target various layers of the application stack that is hosted on the system used to provide the service. These layers include the Operating Systems (OS), server applications such as web servers, DNS servers, databases, and the (typically web-based) applications that sit on top of them. Attacking each layer requires different techniques that take advantage of bottlenecks that are unique to the respective components. A DoS attack may be generated by a single system or multiple systems spread across the internet, which is commonly referred to as a distributed DoS (DDoS). To perform DoS attacks against endpoint resources, several aspects apply to multiple methods, including IP address spoofing and botnets. Adversaries may use the original IP address of an attacking system, or spoof the source IP address to make the attack traffic more difficult to trace back to the attacking system or to enable reflection. This can

increase the difficulty defenders have in defending against the attack by reducing or eliminating the effectiveness of filtering by the source address on network defense devices. Botnets are commonly used to conduct DDoS attacks against networks and services. Large botnets can generate a significant amount of traffic from systems spread across the global internet. Adversaries may have the resources to build out and control their own botnet infrastructure or may rent time on an existing botnet to conduct an attack. In some of the worst cases for DDoS, so many systems are used to generate requests that each one only needs to send out a small amount of traffic to produce enough volume to exhaust the target's resources. In such circumstances, distinguishing DDoS traffic from legitimate clients becomes exceedingly difficult. Botnets have been used in some of the most high-profile DDoS attacks, such as the 2012 series of incidents that targeted major US banks.(Citation: USNYAG IranianBotnet March 2016) In cases where traffic manipulation is used, there may be points in the global network (such as high traffic gateway routers) where packets can be altered and cause legitimate clients to execute code that directs network packets toward a target in high volume. This type of capability was previously used for the purposes of web censorship where client HTTP traffic was modified to include a reference to JavaScript that generated the DDoS code to overwhelm target web servers.(Citation: ArsTechnica Great Firewall of China) For attacks attempting to saturate the providing network, see [Network Denial of Service](<https://attack.mitre.org/techniques/T1498>).

Name

Web Service

ID

T1102

Description

Adversaries may use an existing, legitimate external Web service as a means for relaying data to/from a compromised system. Popular websites and social media acting as a mechanism for C2 may give a significant amount of cover due to the likelihood that hosts within a network are already communicating with them prior to a compromise. Using common services, such as those offered by Google or Twitter, makes it easier for adversaries to hide in expected noise. Web service providers commonly use SSL/TLS encryption, giving adversaries an added level of protection. Use of Web services may also protect back-end C2 infrastructure from discovery through malware binary analysis while

also enabling operational resiliency (since this infrastructure may be dynamically changed).

Name

Indicator Removal

ID

T1070

Description

Adversaries may delete or modify artifacts generated within systems to remove evidence of their presence or hinder defenses. Various artifacts may be created by an adversary or something that can be attributed to an adversary's actions. Typically these artifacts are used as defensive indicators related to monitored events, such as strings from downloaded files, logs that are generated from user actions, and other data analyzed by defenders. Location, format, and type of artifact (such as command or login history) are often specific to each platform. Removal of these indicators may interfere with event collection, reporting, or other processes used to detect intrusion activity. This may compromise the integrity of security solutions by causing notable events to go unreported. This activity may also impede forensic analysis and incident response, due to lack of sufficient data to determine what occurred.

Name

Command and Scripting Interpreter

ID

T1059

Description

Adversaries may abuse command and script interpreters to execute commands, scripts, or binaries. These interfaces and languages provide ways of interacting with computer systems and are a common feature across many different platforms. Most systems come

with some built-in command-line interface and scripting capabilities, for example, macOS and Linux distributions include some flavor of [Unix Shell](<https://attack.mitre.org/techniques/T1059/004>) while Windows installations include the [Windows Command Shell](<https://attack.mitre.org/techniques/T1059/003>) and [PowerShell](<https://attack.mitre.org/techniques/T1059/001>). There are also cross-platform interpreters such as [Python](<https://attack.mitre.org/techniques/T1059/006>), as well as those commonly associated with client applications such as [JavaScript](<https://attack.mitre.org/techniques/T1059/007>) and [Visual Basic](<https://attack.mitre.org/techniques/T1059/005>). Adversaries may abuse these technologies in various ways as a means of executing arbitrary commands. Commands and scripts can be embedded in [Initial Access](<https://attack.mitre.org/tactics/TA0001>) payloads delivered to victims as lure documents or as secondary payloads downloaded from an existing C2. Adversaries may also execute commands through interactive terminals/shells, as well as utilize various [Remote Services](<https://attack.mitre.org/techniques/T1021>) in order to achieve remote Execution. (Citation: Powershell Remote Commands)(Citation: Cisco IOS Software Integrity Assurance - Command History)(Citation: Remote Shell Execution in Python)

Name
Obfuscated Files or Information
ID
T1027
Description
Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit. This is common behavior that can be used across different platforms and the network to evade defenses. Payloads may be compressed, archived, or encrypted in order to avoid detection. These payloads may be used during Initial Access or later to mitigate detection. Sometimes a user's action may be required to open and [Deobfuscate/Decode Files or Information](https://attack.mitre.org/techniques/T1140) for [User Execution](https://attack.mitre.org/techniques/T1204). The user may also be required to input a password to open a password protected compressed/encrypted file that was provided by the adversary. (Citation: Volexity PowerDuke November 2016) Adversaries may also use compressed or archived scripts, such as JavaScript. Portions of files can also be encoded to hide the plain-text strings that would otherwise help defenders with discovery. (Citation: Linux/Cdorked.A We Live Security Analysis) Payloads may also be split into separate, seemingly benign files that only reveal malicious functionality when reassembled. (Citation: Carbon

Black Obfuscation Sept 2016) Adversaries may also abuse [Command Obfuscation](<https://attack.mitre.org/techniques/T1027/010>) to obscure commands executed from payloads or directly via [Command and Scripting Interpreter](<https://attack.mitre.org/techniques/T1059>). Environment variables, aliases, characters, and other platform/language specific semantics can be used to evade signature based detections and application control mechanisms. (Citation: FireEye Obfuscation June 2017) (Citation: FireEye Revoke-Obfuscation July 2017)(Citation: PaloAlto EncodedCommand March 2017)

Name

Native API

ID

T1106

Description

Adversaries may interact with the native OS application programming interface (API) to execute behaviors. Native APIs provide a controlled means of calling low-level OS services within the kernel, such as those involving hardware/devices, memory, and processes. (Citation: NT API Windows)(Citation: Linux Kernel API) These native APIs are leveraged by the OS during system boot (when other system components are not yet initialized) as well as carrying out tasks and requests during routine operations. Adversaries may abuse these OS API functions as a means of executing behaviors. Similar to [Command and Scripting Interpreter](<https://attack.mitre.org/techniques/T1059>), the native API and its hierarchy of interfaces provide mechanisms to interact with and utilize various components of a victimized system. Native API functions (such as `NtCreateProcess`) may be directed invoked via system calls / syscalls, but these features are also often exposed to user-mode applications via interfaces and libraries.(Citation: OutFlank System Calls)(Citation: CyberBit System Calls)(Citation: MDSec System Calls) For example, functions such as the Windows API `CreateProcess()` or GNU `fork()` will allow programs and scripts to start other processes.(Citation: Microsoft CreateProcess)(Citation: GNU Fork) This may allow API callers to execute a binary, run a CLI command, load modules, etc. as thousands of similar API functions exist for various system operations.(Citation: Microsoft Win32)(Citation: LIBC) (Citation: GLIBC) Higher level software frameworks, such as Microsoft .NET and macOS Cocoa, are also available to interact with native APIs. These frameworks typically provide language wrappers/abstractions to API functionalities and are designed for ease-of-use/portability of code.(Citation: Microsoft NET)(Citation: Apple Core Services)(Citation: MACOS Cocoa)(Citation: macOS Foundation) Adversaries may use assembly to directly or indirectly invoke syscalls in an attempt to subvert defensive sensors and detection

signatures such as user mode API-hooks.(Citation: Redops Syscalls) Adversaries may also attempt to tamper with sensors and defensive tools associated with API monitoring, such as unhooking monitored functions via [Disable or Modify Tools](<https://attack.mitre.org/techniques/T1562/001>).

Name

Scheduled Task/Job

ID

T1053

Description

Adversaries may abuse task scheduling functionality to facilitate initial or recurring execution of malicious code. Utilities exist within all major operating systems to schedule programs or scripts to be executed at a specified date and time. A task can also be scheduled on a remote system, provided the proper authentication is met (ex: RPC and file and printer sharing in Windows environments). Scheduling a task on a remote system typically may require being a member of an admin or otherwise privileged group on the remote system.(Citation: TechNet Task Scheduler Security) Adversaries may use task scheduling to execute programs at system startup or on a scheduled basis for persistence. These mechanisms can also be abused to run a process under the context of a specified account (such as one with elevated permissions/privileges). Similar to [System Binary Proxy Execution](<https://attack.mitre.org/techniques/T1218>), adversaries have also abused task scheduling to potentially mask one-time execution under a trusted system process. (Citation: ProofPoint Serpent)

Name

Network Sniffing

ID

T1040

Description

Adversaries may sniff network traffic to capture information about an environment, including authentication material passed over the network. Network sniffing refers to using the network interface on a system to monitor or capture information sent over a wired or wireless connection. An adversary may place a network interface into promiscuous mode to passively access data in transit over the network, or use span ports to capture a larger amount of data. Data captured via this technique may include user credentials, especially those sent over an insecure, unencrypted protocol. Techniques for name service resolution poisoning, such as [LLMNR/NBT-NS Poisoning and SMB Relay] (<https://attack.mitre.org/techniques/T1557/001>), can also be used to capture credentials to websites, proxies, and internal systems by redirecting traffic to an adversary. Network sniffing may also reveal configuration details, such as running services, version numbers, and other network characteristics (e.g. IP addresses, hostnames, VLAN IDs) necessary for subsequent Lateral Movement and/or Defense Evasion activities. In cloud-based environments, adversaries may still be able to use traffic mirroring services to sniff network traffic from virtual machines. For example, AWS Traffic Mirroring, GCP Packet Mirroring, and Azure vTap allow users to define specified instances to collect traffic from and specified targets to send collected traffic to.(Citation: AWS Traffic Mirroring)(Citation: GCP Packet Mirroring)(Citation: Azure Virtual Network TAP) Often, much of this traffic will be in cleartext due to the use of TLS termination at the load balancer level to reduce the strain of encrypting and decrypting traffic.(Citation: Rhino Security Labs AWS VPC Traffic Mirroring)(Citation: SpecterOps AWS Traffic Mirroring) The adversary can then use exfiltration techniques such as Transfer Data to Cloud Account in order to access the snuffed traffic.(Citation: Rhino Security Labs AWS VPC Traffic Mirroring) On network devices, adversaries may perform network captures using [Network Device CLI](<https://attack.mitre.org/techniques/T1059/008>) commands such as `monitor capture`.(Citation: US-CERT-TA18-106A)(Citation: capture_embedded_packet_on_software)

Name

TA0011

ID

TA0011

Name

Network Denial of Service

ID

T1498

Description

Adversaries may perform Network Denial of Service (DoS) attacks to degrade or block the availability of targeted resources to users. Network DoS can be performed by exhausting the network bandwidth services rely on. Example resources include specific websites, email services, DNS, and web-based applications. Adversaries have been observed conducting network DoS attacks for political purposes(Citation: FireEye OpPoisonedHandover February 2016) and to support other malicious activities, including distraction(Citation: FSISAC FraudNetDoS September 2012), hacktivism, and extortion. (Citation: Symantec DDoS October 2014) A Network DoS will occur when the bandwidth capacity of the network connection to a system is exhausted due to the volume of malicious traffic directed at the resource or the network connections and network devices the resource relies on. For example, an adversary may send 10Gbps of traffic to a server that is hosted by a network with a 1Gbps connection to the internet. This traffic can be generated by a single system or multiple systems spread across the internet, which is commonly referred to as a distributed DoS (DDoS). To perform Network DoS attacks several aspects apply to multiple methods, including IP address spoofing, and botnets. Adversaries may use the original IP address of an attacking system, or spoof the source IP address to make the attack traffic more difficult to trace back to the attacking system or to enable reflection. This can increase the difficulty defenders have in defending against the attack by reducing or eliminating the effectiveness of filtering by the source address on network defense devices. For DoS attacks targeting the hosting system directly, see [Endpoint Denial of Service](<https://attack.mitre.org/techniques/T1499>).

Name

Endpoint Denial of Service

ID

T1499

Description

Adversaries may perform Endpoint Denial of Service (DoS) attacks to degrade or block the availability of services to users. Endpoint DoS can be performed by exhausting the system resources those services are hosted on or exploiting the system to cause a persistent crash condition. Example services include websites, email services, DNS, and web-based applications. Adversaries have been observed conducting DoS attacks for political purposes(Citation: FireEye OpPoisonedHandover February 2016) and to support other malicious activities, including distraction(Citation: FSISAC FraudNetDoS September 2012), hacktivism, and extortion.(Citation: Symantec DDoS October 2014) An Endpoint DoS denies the availability of a service without saturating the network used to provide access to the service. Adversaries can target various layers of the application stack that is hosted on the system used to provide the service. These layers include the Operating Systems (OS), server applications such as web servers, DNS servers, databases, and the (typically web-based) applications that sit on top of them. Attacking each layer requires different techniques that take advantage of bottlenecks that are unique to the respective components. A DoS attack may be generated by a single system or multiple systems spread across the internet, which is commonly referred to as a distributed DoS (DDoS). To perform DoS attacks against endpoint resources, several aspects apply to multiple methods, including IP address spoofing and botnets. Adversaries may use the original IP address of an attacking system, or spoof the source IP address to make the attack traffic more difficult to trace back to the attacking system or to enable reflection. This can increase the difficulty defenders have in defending against the attack by reducing or eliminating the effectiveness of filtering by the source address on network defense devices. Botnets are commonly used to conduct DDoS attacks against networks and services. Large botnets can generate a significant amount of traffic from systems spread across the global internet. Adversaries may have the resources to build out and control their own botnet infrastructure or may rent time on an existing botnet to conduct an attack. In some of the worst cases for DDoS, so many systems are used to generate requests that each one only needs to send out a small amount of traffic to produce enough volume to exhaust the target's resources. In such circumstances, distinguishing DDoS traffic from legitimate clients becomes exceedingly difficult. Botnets have been used in some of the most high-profile DDoS attacks, such as the 2012 series of incidents that targeted major US banks.(Citation: USNYAG IranianBotnet March 2016) In cases where traffic manipulation is used, there may be points in the global network (such as high traffic gateway routers) where packets can be altered and cause legitimate clients to execute code that directs network packets toward a target in high volume. This type of capability was previously used for the purposes of web censorship where client HTTP traffic was modified to include a reference to JavaScript that generated the DDoS code to overwhelm target web servers.(Citation: ArsTechnica Great Firewall of China) For attacks attempting to saturate the providing network, see [Network Denial of Service](<https://attack.mitre.org/techniques/T1498>).

Name

Web Service

ID

T1102

Description

Adversaries may use an existing, legitimate external Web service as a means for relaying data to/from a compromised system. Popular websites and social media acting as a mechanism for C2 may give a significant amount of cover due to the likelihood that hosts within a network are already communicating with them prior to a compromise. Using common services, such as those offered by Google or Twitter, makes it easier for adversaries to hide in expected noise. Web service providers commonly use SSL/TLS encryption, giving adversaries an added level of protection. Use of Web services may also protect back-end C2 infrastructure from discovery through malware binary analysis while also enabling operational resiliency (since this infrastructure may be dynamically changed).

Name

Indicator Removal

ID

T1070

Description

Adversaries may delete or modify artifacts generated within systems to remove evidence of their presence or hinder defenses. Various artifacts may be created by an adversary or something that can be attributed to an adversary's actions. Typically these artifacts are used as defensive indicators related to monitored events, such as strings from downloaded files, logs that are generated from user actions, and other data analyzed by defenders. Location, format, and type of artifact (such as command or login history) are often specific to each platform. Removal of these indicators may interfere with event collection, reporting, or other processes used to detect intrusion activity. This may compromise the integrity of security solutions by causing notable events to go unreported.

This activity may also impede forensic analysis and incident response, due to lack of sufficient data to determine what occurred.

Name

Command and Scripting Interpreter

ID

T1059

Description

Adversaries may abuse command and script interpreters to execute commands, scripts, or binaries. These interfaces and languages provide ways of interacting with computer systems and are a common feature across many different platforms. Most systems come with some built-in command-line interface and scripting capabilities, for example, macOS and Linux distributions include some flavor of [Unix Shell](<https://attack.mitre.org/techniques/T1059/004>) while Windows installations include the [Windows Command Shell](<https://attack.mitre.org/techniques/T1059/003>) and [PowerShell](<https://attack.mitre.org/techniques/T1059/001>). There are also cross-platform interpreters such as [Python](<https://attack.mitre.org/techniques/T1059/006>), as well as those commonly associated with client applications such as [JavaScript](<https://attack.mitre.org/techniques/T1059/007>) and [Visual Basic](<https://attack.mitre.org/techniques/T1059/005>). Adversaries may abuse these technologies in various ways as a means of executing arbitrary commands. Commands and scripts can be embedded in [Initial Access](<https://attack.mitre.org/tactics/TA0001>) payloads delivered to victims as lure documents or as secondary payloads downloaded from an existing C2. Adversaries may also execute commands through interactive terminals/shells, as well as utilize various [Remote Services](<https://attack.mitre.org/techniques/T1021>) in order to achieve remote Execution. (Citation: Powershell Remote Commands)(Citation: Cisco IOS Software Integrity Assurance - Command History)(Citation: Remote Shell Execution in Python)

Name

Obfuscated Files or Information

ID

T1027

Description

Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit. This is common behavior that can be used across different platforms and the network to evade defenses. Payloads may be compressed, archived, or encrypted in order to avoid detection. These payloads may be used during Initial Access or later to mitigate detection. Sometimes a user's action may be required to open and [Deobfuscate/Decode Files or Information](<https://attack.mitre.org/techniques/T1140>) for [User Execution](<https://attack.mitre.org/techniques/T1204>). The user may also be required to input a password to open a password protected compressed/encrypted file that was provided by the adversary. (Citation: Volexity PowerDuke November 2016) Adversaries may also use compressed or archived scripts, such as JavaScript. Portions of files can also be encoded to hide the plain-text strings that would otherwise help defenders with discovery. (Citation: Linux/Cdorked.A We Live Security Analysis) Payloads may also be split into separate, seemingly benign files that only reveal malicious functionality when reassembled. (Citation: Carbon Black Obfuscation Sept 2016) Adversaries may also abuse [Command Obfuscation](<https://attack.mitre.org/techniques/T1027/010>) to obscure commands executed from payloads or directly via [Command and Scripting Interpreter](<https://attack.mitre.org/techniques/T1059>). Environment variables, aliases, characters, and other platform/language specific semantics can be used to evade signature based detections and application control mechanisms. (Citation: FireEye Obfuscation June 2017) (Citation: FireEye Revoke-Obfuscation July 2017)(Citation: PaloAlto EncodedCommand March 2017)

Name

Native API

ID

T1106

Description

Adversaries may interact with the native OS application programming interface (API) to execute behaviors. Native APIs provide a controlled means of calling low-level OS services within the kernel, such as those involving hardware/devices, memory, and processes.

(Citation: NT API Windows)(Citation: Linux Kernel API) These native APIs are leveraged by the OS during system boot (when other system components are not yet initialized) as well as carrying out tasks and requests during routine operations. Adversaries may abuse these OS API functions as a means of executing behaviors. Similar to [Command and Scripting Interpreter](<https://attack.mitre.org/techniques/T1059>), the native API and its hierarchy of interfaces provide mechanisms to interact with and utilize various components of a victimized system. Native API functions (such as `NtCreateProcess`) may be directed invoked via system calls / syscalls, but these features are also often exposed to user-mode applications via interfaces and libraries.(Citation: OutFlank System Calls)(Citation: CyberBit System Calls)(Citation: MDSec System Calls) For example, functions such as the Windows API `CreateProcess()` or GNU `fork()` will allow programs and scripts to start other processes.(Citation: Microsoft CreateProcess)(Citation: GNU Fork) This may allow API callers to execute a binary, run a CLI command, load modules, etc. as thousands of similar API functions exist for various system operations.(Citation: Microsoft Win32)(Citation: LIBC) (Citation: GLIBC) Higher level software frameworks, such as Microsoft .NET and macOS Cocoa, are also available to interact with native APIs. These frameworks typically provide language wrappers/abstractions to API functionalities and are designed for ease-of-use/portability of code.(Citation: Microsoft .NET)(Citation: Apple Core Services)(Citation: MACOS Cocoa)(Citation: macOS Foundation) Adversaries may use assembly to directly or indirectly invoke syscalls in an attempt to subvert defensive sensors and detection signatures such as user mode API-hooks.(Citation: Redops Syscalls) Adversaries may also attempt to tamper with sensors and defensive tools associated with API monitoring, such as unhooking monitored functions via [Disable or Modify Tools](<https://attack.mitre.org/techniques/T1562/001>).

Name

Scheduled Task/Job

ID

T1053

Description

Adversaries may abuse task scheduling functionality to facilitate initial or recurring execution of malicious code. Utilities exist within all major operating systems to schedule programs or scripts to be executed at a specified date and time. A task can also be scheduled on a remote system, provided the proper authentication is met (ex: RPC and file and printer sharing in Windows environments). Scheduling a task on a remote system typically may require being a member of an admin or otherwise privileged group on the

remote system.(Citation: TechNet Task Scheduler Security) Adversaries may use task scheduling to execute programs at system startup or on a scheduled basis for persistence. These mechanisms can also be abused to run a process under the context of a specified account (such as one with elevated permissions/privileges). Similar to [System Binary Proxy Execution](<https://attack.mitre.org/techniques/T1218>), adversaries have also abused task scheduling to potentially mask one-time execution under a trusted system process. (Citation: ProofPoint Serpent)

Name

Network Sniffing

ID

T1040

Description

Adversaries may sniff network traffic to capture information about an environment, including authentication material passed over the network. Network sniffing refers to using the network interface on a system to monitor or capture information sent over a wired or wireless connection. An adversary may place a network interface into promiscuous mode to passively access data in transit over the network, or use span ports to capture a larger amount of data. Data captured via this technique may include user credentials, especially those sent over an insecure, unencrypted protocol. Techniques for name service resolution poisoning, such as [LLMNR/NBT-NS Poisoning and SMB Relay](<https://attack.mitre.org/techniques/T1557/001>), can also be used to capture credentials to websites, proxies, and internal systems by redirecting traffic to an adversary. Network sniffing may also reveal configuration details, such as running services, version numbers, and other network characteristics (e.g. IP addresses, hostnames, VLAN IDs) necessary for subsequent Lateral Movement and/or Defense Evasion activities. In cloud-based environments, adversaries may still be able to use traffic mirroring services to sniff network traffic from virtual machines. For example, AWS Traffic Mirroring, GCP Packet Mirroring, and Azure vTap allow users to define specified instances to collect traffic from and specified targets to send collected traffic to.(Citation: AWS Traffic Mirroring)(Citation: GCP Packet Mirroring)(Citation: Azure Virtual Network TAP) Often, much of this traffic will be in cleartext due to the use of TLS termination at the load balancer level to reduce the strain of encrypting and decrypting traffic.(Citation: Rhino Security Labs AWS VPC Traffic Mirroring)(Citation: SpecterOps AWS Traffic Mirroring) The adversary can then use exfiltration techniques such as Transfer Data to Cloud Account in order to access the sniffed traffic.(Citation: Rhino Security Labs AWS VPC Traffic Mirroring) On network devices,

adversaries may perform network captures using [Network Device CLI](<https://attack.mitre.org/techniques/T1059/008>) commands such as `monitor capture`.(Citation: US-CERT-TA18-106A)(Citation: capture_embedded_packet_on_software)

StixFile

Value
9ebcac15e36dd79439d5f92feb1933e0a68e75cc1e9eecb6369e4b3317e2949
9ebcac15e36dd79439d5f92feb1933e0a68e75cc1e9eecb6369e4b3317e2949

IPv4-Addr

Value
94.103.188.167
45.95.146.93
185.132.125.193
94.103.188.167
45.95.146.93
185.132.125.193

External References

- https://blog.xlab.qianxin.com/smargaft_abusing_binance-smart-contracts_en/
- <https://otx.alienvault.com/pulse/65c0f0f561a3fcda5f097b3e>
