Jun 11 2024

NETMANAGEIT

Intelligence Report

New Agent Tesla Campaign Targeting Spanish-Speaking People

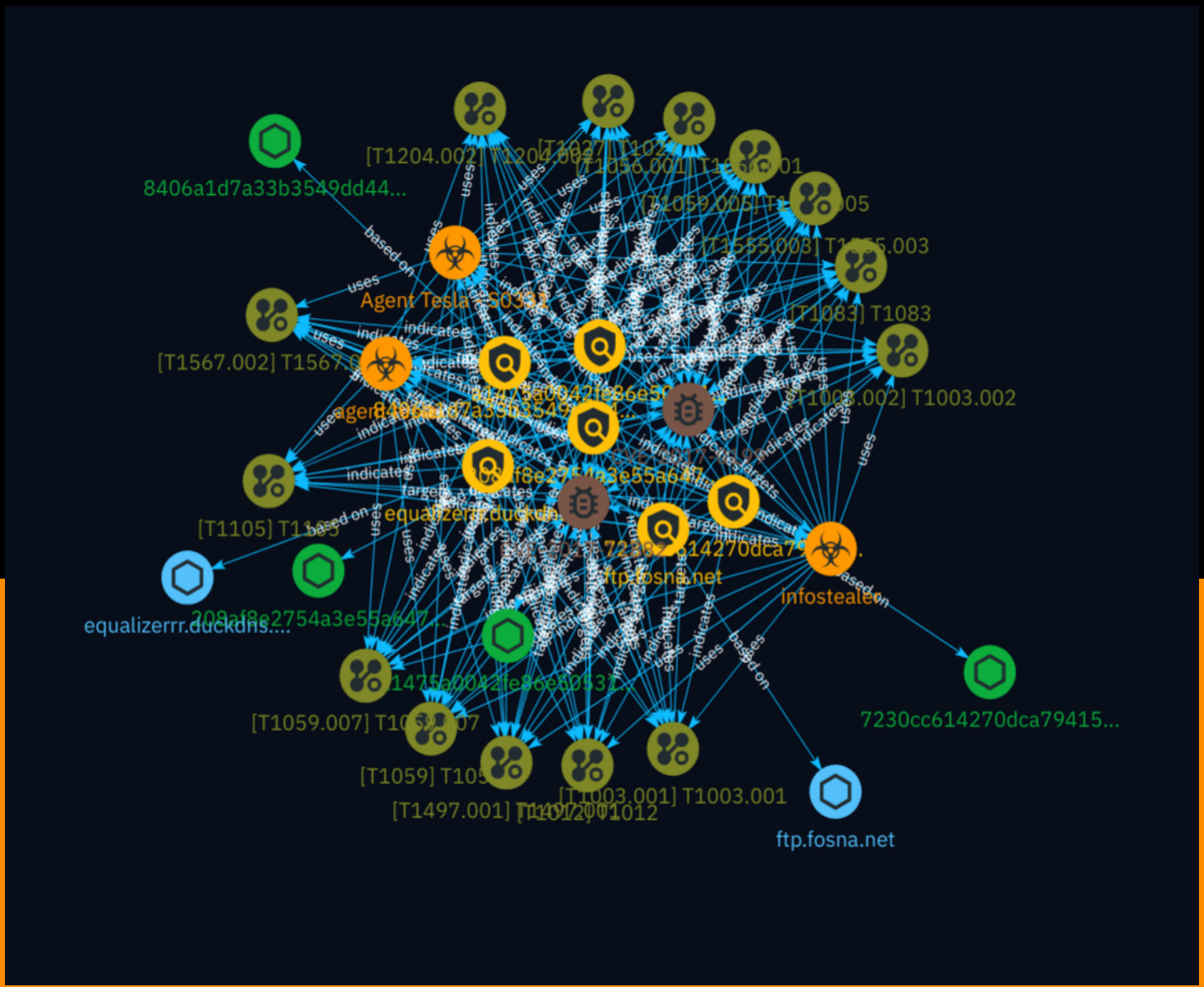

Table of contents

Overview

Entities

Observables

External References

Overview

Description

This report analyzes a phishing campaign spreading a new Agent Tesla variant designed to infiltrate victims' computers and steal sensitive information like credentials, email contacts, and system details. It leverages techniques like exploiting Microsoft Office vulnerabilities, executing JavaScript and PowerShell code, process hollowing, and obfuscation to evade detection. The malware targets over 80 software applications to harvest credentials and collects email contacts from Thunderbird. Stolen data is exfiltrated via FTP. Fortinet's security services provide protection against this campaign.

Confidence

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

100 / 100

Content

N/A

Attack-Pattern

Name
T1012
ID
T1012
Description
Adversaries may interact with the Windows Registry to gather information about the system, configuration, and installed software. The Registry contains a significant amount of information about the operating system, configuration, software, and security.(Citation: Wikipedia Windows Registry) Information can easily be queried using the [Reg](https://attack.mitre.org/software/S0075) utility, though other means to access the Registry exist. Some of the information may help adversaries to further their operation within a network. Adversaries may use the information from [Query Registry](https://attack.mitre.org/techniques/T1012) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.
Name
T1555.003
ID
T1555.003

Description

Adversaries may acquire credentials from web browsers by reading files specific to the target browser.(Citation: Talos Olympic Destroyer 2018) Web browsers commonly save credentials such as website usernames and passwords so that they do not need to be entered manually in the future. Web browsers typically store the credentials in an encrypted format within a credential store; however, methods exist to extract plaintext credentials from web browsers. For example, on Windows systems, encrypted credentials may be obtained from Google Chrome by reading a database file, `AppData\Local\Google\Chrome\User Data\Default\Login Data` and executing a SQL query: `SELECT action_url, username_value, password_value FROM logins;`. The plaintext password can then be obtained by passing the encrypted credentials to the Windows API function `CryptUnprotectData`, which uses the victim's cached logon credentials as the decryption key.(Citation: Microsoft CryptUnprotectData April 2018) Adversaries have executed similar procedures for common web browsers such as FireFox, Safari, Edge, etc. (Citation: Proofpoint Vega Credential Stealer May 2018)(Citation: FireEye HawkEye Malware July 2017) Windows stores Internet Explorer and Microsoft Edge credentials in Credential Lockers managed by the [Windows Credential Manager](https://attack.mitre.org/techniques/T1555/004). Adversaries may also acquire credentials by searching web browser process memory for patterns that commonly match credentials.(Citation: GitHub Mimikittenz July 2016) After acquiring credentials from web browsers, adversaries may attempt to recycle the credentials across different systems and/or accounts in order to expand access. This can result in significantly furthering an adversary's objective in cases where credentials gained from web browsers overlap with privileged accounts (e.g. domain administrator).

Name

T1059.007

ID

T1059.007

Description

Adversaries may abuse various implementations of JavaScript for execution. JavaScript (JS) is a platform-independent scripting language (compiled just-in-time at runtime) commonly associated with scripts in webpages, though JS can be executed in runtime environments outside the browser.(Citation: NodeJS) JScript is the Microsoft

implementation of the same scripting standard. JScript is interpreted via the Windows Script engine and thus integrated with many components of Windows such as the [Component Object Model](https://attack.mitre.org/techniques/T1559/001) and Internet Explorer HTML Application (HTA) pages.(Citation: JScrip May 2018)(Citation: Microsoft JScript 2007)(Citation: Microsoft Windows Scripts) JavaScript for Automation (JXA) is a macOS scripting language based on JavaScript, included as part of Apple's Open Scripting Architecture (OSA), that was introduced in OSX 10.10. Apple's OSA provides scripting capabilities to control applications, interface with the operating system, and bridge access into the rest of Apple's internal APIs. As of OSX 10.10, OSA only supports two languages, JXA and [AppleScript](https://attack.mitre.org/techniques/T1059/002). Scripts can be executed via the command line utility `osascript`, they can be compiled into applications or script files via `osacompile`, and they can be compiled and executed in memory of other programs by leveraging the OSAKit Framework.(Citation: Apple About Mac Scripting 2016)(Citation: SpecterOps JXA 2020)(Citation: SentinelOne macOS Red Team)(Citation: Red Canary Silver Sparrow Feb2021)(Citation: MDSec macOS JXA and VSCode) Adversaries may abuse various implementations of JavaScript to execute various behaviors. Common uses include hosting malicious scripts on websites as part of a [Drive-by Compromise](https://attack.mitre.org/techniques/T1189) or downloading and executing these script files as secondary payloads. Since these payloads are text-based, it is also very common for adversaries to obfuscate their content as part of [Obfuscated Files or Information](https://attack.mitre.org/techniques/T1027).

Name

T1003.002

ID

T1003.002

Description

Adversaries may attempt to extract credential material from the Security Account Manager (SAM) database either through in-memory techniques or through the Windows Registry where the SAM database is stored. The SAM is a database file that contains local accounts for the host, typically those found with the `net user` command. Enumerating the SAM database requires SYSTEM level access. A number of tools can be used to retrieve the SAM file through in-memory techniques: * pwdumpx.exe * [gsecdump](https://attack.mitre.org/software/S0008) * [Mimikatz](https://attack.mitre.org/software/S0002) * secretsdump.py Alternatively, the SAM can be extracted from the Registry with Reg: * `reg save HKLM\sam sam` * `reg save HKLM\system system` Creddump7 can then be used to process the SAM

database locally to retrieve hashes.(Citation: GitHub Creddump7) Notes: * RID 500 account is the local, built-in administrator. * RID 501 is the guest account. * User accounts start with a RID of 1,000+.

Name

T1056.001

ID

T1056.001

Description

Adversaries may log user keystrokes to intercept credentials as the user types them. Keylogging is likely to be used to acquire credentials for new access opportunities when [OS Credential Dumping](https://attack.mitre.org/techniques/T1003) efforts are not effective, and may require an adversary to intercept keystrokes on a system for a substantial period of time before credentials can be successfully captured. In order to increase the likelihood of capturing credentials quickly, an adversary may also perform actions such as clearing browser cookies to force users to reauthenticate to systems. (Citation: Talos Kimsuky Nov 2021) Keylogging is the most prevalent type of input capture, with many different ways of intercepting keystrokes.(Citation: Adventures of a Keystroke) Some methods include: * Hooking API callbacks used for processing keystrokes. Unlike [Credential API Hooking](https://attack.mitre.org/techniques/T1056/004), this focuses solely on API functions intended for processing keystroke data. * Reading raw keystroke data from the hardware buffer. * Windows Registry modifications. * Custom drivers. * [Modify System Image](https://attack.mitre.org/techniques/T1601) may provide adversaries with hooks into the operating system of network devices to read raw keystrokes for login sessions.(Citation: Cisco Blog Legacy Device Attacks)

Name

T1204.002

ID

T1204.002

Description

An adversary may rely upon a user opening a malicious file in order to gain execution. Users may be subjected to social engineering to get them to open a file that will lead to code execution. This user action will typically be observed as follow-on behavior from [Spearphishing Attachment](https://attack.mitre.org/techniques/T1566/001). Adversaries may use several types of files that require a user to execute them, including .doc, .pdf, .xls, .rtf, .scr, .exe, .lnk, .pif, and .cpl. Adversaries may employ various forms of [Masquerading](https://attack.mitre.org/techniques/T1036) and [Obfuscated Files or Information](https://attack.mitre.org/techniques/T1027) to increase the likelihood that a user will open and successfully execute a malicious file. These methods may include using a familiar naming convention and/or password protecting the file and supplying instructions to a user on how to open it.(Citation: Password Protected Word Docs) While [Malicious File](https://attack.mitre.org/techniques/T1204/002) frequently occurs shortly after Initial Access it may occur at other phases of an intrusion, such as when an adversary places a file in a shared directory or on a user's desktop hoping that a user will click on it. This activity may also be seen shortly after [Internal Spearphishing](https://attack.mitre.org/techniques/T1534).

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

T1567.002

ID

T1567.002

Description

Adversaries may exfiltrate data to a cloud storage service rather than over their primary command and control channel. Cloud storage services allow for the storage, edit, and retrieval of data from a remote cloud storage server over the Internet. Examples of cloud storage services include Dropbox and Google Docs. Exfiltration to these cloud storage services can provide a significant amount of cover to the adversary if hosts within the network are already communicating with the service.

Name

T1105

ID

T1105

Description

Adversaries may transfer tools or other files from an external system into a compromised environment. Tools or files may be copied from an external adversary-controlled system to the victim network through the command and control channel or through alternate protocols such as [ftp](https://attack.mitre.org/software/S0095). Once present, adversaries may also transfer/spread tools between victim devices within a compromised environment (i.e. [Lateral Tool Transfer](https://attack.mitre.org/techniques/T1570)). On Windows, adversaries may use various utilities to download tools, such as `copy`, `finger`, [certutil](https://attack.mitre.org/software/S0160), and [PowerShell](https://attack.mitre.org/techniques/T1059/001) commands such as `IEX(New-Object Net.WebClient).downloadString()` and `Invoke-WebRequest`. On Linux and macOS systems, a variety of utilities also exist, such as `curl`, `scp`, `sftp`, `tftp`, `rsync`, `finger`, and `wget`. (Citation: t1105_lolbas) Adversaries may also abuse installers and package managers, such as `yum` or `winget`, to download tools to victim hosts. Files can also be transferred using various [Web Service](https://attack.mitre.org/techniques/T1102)s as well as native or otherwise present tools on the victim system.(Citation: PTSecurity Cobalt Dec 2016) In some cases, adversaries may be able to leverage services that sync between a web-based and an on-premises client, such as Dropbox or OneDrive, to transfer files onto victim systems. For example, by compromising a cloud account and logging into the service's web portal, an adversary may be able to trigger an automatic syncing process that transfers the file onto the victim's machine.(Citation: Dropbox Malware Sync)

Name

T1003.001

ID

T1003.001

Description

Adversaries may attempt to access credential material stored in the process memory of the Local Security Authority Subsystem Service (LSASS). After a user logs on, the system generates and stores a variety of credential materials in LSASS process memory. These credential materials can be harvested by an administrative user or SYSTEM and used to conduct [Lateral Movement](https://attack.mitre.org/tactics/TA0008) using [Use Alternate Authentication Material](https://attack.mitre.org/techniques/T1550). As well as in-memory techniques, the LSASS process memory can be dumped from the target host and analyzed on a local system. For example, on the target host use procdump: * `procdump -ma lsass.exe lsass_dump` Locally, mimikatz can be run using: * `sekurlsa::Minidump

lsassdump.dmp` * `sekurlsa::logonPasswords` Built-in Windows tools such as comsvcs.dll can also be used: * `rundll32.exe C:\Windows\System32\comsvcs.dll MiniDump PID lsass.dmp full`(Citation: Volexity Exchange Marauder March 2021)(Citation: Symantec Attacks Against Government Sector) Windows Security Support Provider (SSP) DLLs are loaded into LSASS process at system start. Once loaded into the LSA, SSP DLLs have access to encrypted and plaintext passwords that are stored in Windows, such as any logged-on user's Domain password or smart card PINs. The SSP configuration is stored in two Registry keys: `HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages` and `HKLM\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\Security Packages`. An adversary may modify these Registry keys to add new SSPs, which will be loaded the next time the system boots, or when the AddSecurityPackage Windows API function is called.(Citation: Graeber 2014) The following SSPs can be used to access credentials: * Msv: Interactive logons, batch logons, and service logons are done through the MSV authentication package. * Wdigest: The Digest Authentication protocol is designed for use with Hypertext Transfer Protocol (HTTP) and Simple Authentication Security Layer (SASL) exchanges. (Citation: TechNet Blogs Credential Protection) * Kerberos: Preferred for mutual client-server domain authentication in Windows 2000 and later. * CredSSP: Provides SSO and Network Level Authentication for Remote Desktop Services.(Citation: TechNet Blogs Credential Protection)

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

T1059.005

ID

T1059.005

Description

Adversaries may abuse Visual Basic (VB) for execution. VB is a programming language created by Microsoft with interoperability with many Windows technologies such as [Component Object Model](https://attack.mitre.org/techniques/T1559/001) and the [Native API](https://attack.mitre.org/techniques/T1106) through the Windows API. Although tagged as legacy with no planned future evolutions, VB is integrated and supported in the .NET Framework and cross-platform .NET Core.(Citation: VB .NET Mar 2020)(Citation: VB Microsoft) Derivative languages based on VB have also been created, such as Visual Basic for Applications (VBA) and VBScript. VBA is an event-driven programming language built into Microsoft Office, as well as several third-party applications.(Citation: Microsoft VBA) (Citation: Wikipedia VBA) VBA enables documents to contain macros used to automate the execution of tasks and other functionality on the host. VBScript is a default scripting language on Windows hosts and can also be used in place of [JavaScript](https://attack.mitre.org/techniques/T1059/007) on HTML Application (HTA) webpages served to Internet Explorer (though most modern browsers do not come with VBScript support). (Citation: Microsoft VBScript) Adversaries may use VB payloads to execute malicious commands. Common malicious usage includes automating execution of behaviors with VBScript or embedding VBA content into [Spearphishing Attachment](https://attack.mitre.org/techniques/T1566/001) payloads (which may also involve [Mark-of-the-Web Bypass](https://attack.mitre.org/techniques/T1553/005) to enable execution).(Citation: Default VBS macros Blocking)

Name

T1497.001

ID

T1497.001

Description

Adversaries may employ various system checks to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads. Adversaries may use the information learned from [Virtualization/Sandbox Evasion](https://attack.mitre.org/techniques/T1497) during automated discovery to shape follow-on behaviors.(Citation: Deloitte Environment Awareness) Specific checks will vary based on the target and/or adversary, but may involve behaviors such as [Windows Management Instrumentation](https://attack.mitre.org/techniques/T1047), [PowerShell](https://attack.mitre.org/techniques/T1059/001), [System Information Discovery](https://attack.mitre.org/techniques/T1082), and [Query Registry](https://attack.mitre.org/techniques/T1012) to obtain system information and search for VME artifacts. Adversaries may search for VME artifacts in memory, processes, file system, hardware, and/or the Registry. Adversaries may use scripting to automate these checks into one script and then have the program exit if it determines the system to be a virtual environment. Checks could include generic system properties such as host/domain name and samples of network traffic. Adversaries may also check the network adapters addresses, CPU core count, and available memory/drive size. Other common checks may enumerate services running that are unique to these applications, installed programs on the system, manufacturer/product fields for strings relating to virtual machine applications, and VME-specific hardware/processor instructions.(Citation: McAfee Virtual Jan 2017) In applications like VMWare, adversaries can also use a special I/O port to send commands and receive output. Hardware checks, such as the presence of the fan, temperature, and audio devices, could also be used to gather evidence that can be indicative a virtual environment. Adversaries may also query for specific readings from these devices.(Citation: Unit 42 OilRig Sept 2018)

Name

T1083

ID

T1083

Description

Adversaries may enumerate files and directories or may search in specific locations of a host or network share for certain information within a file system. Adversaries may use the information from [File and Directory Discovery](https://attack.mitre.org/techniques/T1083) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions. Many command shell utilities can be used to obtain this information. Examples include `dir`, `tree`, `ls`, `find`, and `locate`.(Citation: Windows Commands JPCERT) Custom tools may also be used to gather file and directory information and interact with the [Native API](https://attack.mitre.org/techniques/T1106). Adversaries may also leverage a [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) on network devices to gather file and directory information (e.g. `dir`, `show flash`, and/or `nvram`).(Citation: US-CERT-TA18-106A)

Indicator

Name
208af8e2754a3e55a64796b29ef3a625d89a357c59c43d0ff4d2d30e20092d74
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '208af8e2754a3e55a64796b29ef3a625d89a357c59c43d0ff4d2d30e20092d74']
Name
8406a1d7a33b3549dd44f551e5a68392f85b5ef9cf8f9f3db68bd7e02d1eaba7
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '8406a1d7a33b3549dd44f551e5a68392f85b5ef9cf8f9f3db68bd7e02d1eaba7']
Name

equalizerrr.duckdns.org

Pattern Type

stix

Pattern

[hostname:value = 'equalizerrr.duckdns.org']

Name

a1475a0042fe86e50531bb8b8182f9e27a3a61f204700f42fd26406c3bdec862

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = 'a1475a0042fe86e50531bb8b8182f9e27a3a61f204700f42fd26406c3bdec862']

Name

7230cc614270dca79415b0cf53a666a219beb4beed90c85a1ac09f082aea613b

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '7230cc614270dca79415b0cf53a666a219beb4beed90c85a1ac09f082aea613b']

Name
ftp.fosna.net
Pattern Type
stix
Pattern
[hostname:value = 'ftp.fosna.net']

Malware

Name
infostealer

Name
agent tesla

Description
[Agent Tesla](https://attack.mitre.org/software/S0331) is a spyware Trojan written for the .NET framework that has been observed since at least 2014.(Citation: Fortinet Agent Tesla April 2018)(Citation: Bitdefender Agent Tesla April 2020)(Citation: Malwarebytes Agent Tesla April 2020)

Name
Agent Tesla - S0331

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

uses

Name

Name

Description

[Agent Tesla](https://attack.mitre.org/software/S0331) can log keystrokes on the victim's machine.(Citation: Talos Agent Tesla Oct 2018)(Citation: DigiTrust Agent Tesla Jan 2017) (Citation: Fortinet Agent Tesla June 2017)(Citation: Bitdefender Agent Tesla April 2020) (Citation: SentinelLabs Agent Tesla Aug 2020)

Name

Name

Name

Name

Name

Name
Name
Name
Name
Name
Name
Name
Name
Name
Name
Description

[Agent Tesla](https://attack.mitre.org/software/S0331) has had its code obfuscated in an apparent attempt to make analysis difficult.(Citation: Fortinet Agent Tesla April 2018) [Agent Tesla](https://attack.mitre.org/software/S0331) has used the Rijndael symmetric encryption algorithm to encrypt strings.(Citation: Malwarebytes Agent Tesla April 2020)

Name

Name

targets

Name
Name
Name
Name
Name
Name
Name
Name
Name
Name

Name
Name
Name
Name
Name
Name

based-on

Name

Name

StixFile

Value
8406a1d7a33b3549dd44f551e5a68392f85b5ef9cf8f9f3db68bd7e02d1eaba7
a1475a0042fe86e50531bb8b8182f9e27a3a61f204700f42fd26406c3bdec862
208af8e2754a3e55a64796b29ef3a625d89a357c59c43d0ff4d2d30e20092d74
7230cc614270dca79415b0cf53a666a219beb4beed90c85a1ac09f082aea613b

Hostname

Value
equalizerrr.duckdns.org
ftp.fosna.net

External References

- https://www.fortinet.com/blog/threat-research/new-agent-tesla-campaign-targeting-spanish-speaking-people
- https://otx.alienvault.com/pulse/6666e26a4f2ba710caaa6046