



# Table of contents

---

## Overview

---

● Description	4
● Confidence	4
● Content	5

---

## Entities

---

● Attack-Pattern	6
● Indicator	15
● Malware	17
● indicates	18
● uses	24
● based-on	26

---

## Observables

---

● Domain-Name	27
---------------	----

---

● StixFile	28
● Hostname	29

---

---

## External References

---

● External References	30
-----------------------	----

# Overview

## Description

This analysis covers the distribution of AsyncRAT malware disguised as an ebook. The compressed file contains a malicious LNK and PowerShell scripts that ultimately execute AsyncRAT. The malware employs various techniques, such as obfuscation, task scheduling, and anti-VM and anti-AV capabilities, to maintain persistence and exfiltrate user information while receiving commands from the threat actor. Particular caution is advised due to the potential for widespread distribution via phishing emails and file-sharing websites.

## Confidence

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

100 / 100

# Content

N/A

# Attack-Pattern

## Name

T1057

## ID

T1057

## Description

Adversaries may attempt to get information about running processes on a system. Information obtained could be used to gain an understanding of common software/ applications running on systems within the network. Administrator or otherwise elevated access may provide better process details. Adversaries may use the information from [Process Discovery](<https://attack.mitre.org/techniques/T1057>) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions. In Windows environments, adversaries could obtain details on running processes using the [Tasklist](<https://attack.mitre.org/software/S0057>) utility via [cmd](<https://attack.mitre.org/software/S0106>) or `Get-Process` via [PowerShell](<https://attack.mitre.org/techniques/T1059/001>). Information about processes can also be extracted from the output of [Native API](<https://attack.mitre.org/techniques/T1106>) calls such as `CreateToolhelp32Snapshot`. In Mac and Linux, this is accomplished with the `ps` command. Adversaries may also opt to enumerate processes via `/proc`. On network devices, [Network Device CLI](<https://attack.mitre.org/techniques/T1059/008>) commands such as `show processes` can be used to display current running processes. (Citation: US-CERT-TA18-106A)(Citation: show\_processes\_cisco\_cmd)

## Name

T1497

**ID**

T1497

**Description**

Adversaries may employ various means 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) Adversaries may use several methods to accomplish [Virtualization/Sandbox Evasion](<https://attack.mitre.org/techniques/T1497>) such as checking for security monitoring tools (e.g., Sysinternals, Wireshark, etc.) or other system artifacts associated with analysis or virtualization. Adversaries may also check for legitimate user activity to help determine if it is in an analysis environment. Additional methods include use of sleep timers or loops within malware code to avoid operating within a temporary sandbox. (Citation: Unit 42 Pirpi July 2015)

**Name**

T1548

**ID**

T1548

**Description**

Adversaries may circumvent mechanisms designed to control elevate privileges to gain higher-level permissions. Most modern systems contain native elevation control mechanisms that are intended to limit privileges that a user can perform on a machine. Authorization has to be granted to specific users in order to perform tasks that can be considered of higher risk.(Citation: TechNet How UAC Works)(Citation: sudo man page 2018) An adversary can perform several methods to take advantage of built-in control

mechanisms in order to escalate privileges on a system.(Citation: OSX Keydnap malware)  
(Citation: Fortinet Fareit)

**Name**

T1064

**ID**

T1064

**Description**

\*\*This technique has been deprecated. Please use [Command and Scripting Interpreter] (<https://attack.mitre.org/techniques/T1059>) where appropriate.\*\* Adversaries may use scripts to aid in operations and perform multiple actions that would otherwise be manual. Scripting is useful for speeding up operational tasks and reducing the time required to gain access to critical resources. Some scripting languages may be used to bypass process monitoring mechanisms by directly interacting with the operating system at an API level instead of calling other programs. Common scripting languages for Windows include VBScript and [PowerShell](<https://attack.mitre.org/techniques/T1086>) but could also be in the form of command-line batch scripts. Scripts can be embedded inside Office documents as macros that can be set to execute when files used in [Spearphishing Attachment](<https://attack.mitre.org/techniques/T1193>) and other types of spearphishing are opened. Malicious embedded macros are an alternative means of execution than software exploitation through [Exploitation for Client Execution](<https://attack.mitre.org/techniques/T1203>), where adversaries will rely on macros being allowed or that the user will accept to activate them. Many popular offensive frameworks exist which use forms of scripting for security testers and adversaries alike. Metasploit (Citation: Metasploit\_Ref), Veil (Citation: Veil\_Ref), and PowerSploit (Citation: Powersploit) are three examples that are popular among penetration testers for exploit and post-compromise operations and include many features for evading defenses. Some adversaries are known to use PowerShell. (Citation: Alperovitch 2014)

**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 abusible system utilities to evade security monitoring is also a form of [Masquerading](https://attack.mitre.org/techniques/T1036).(Citation: LOLBAS Main Site)

**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: JScript 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**

T1070

**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**

T1059.001

**ID**

T1059.001

**Description**

Adversaries may abuse PowerShell commands and scripts for execution. PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system.(Citation: TechNet PowerShell) Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Examples include the `Start-Process` cmdlet which can be used to run an executable and the `Invoke-Command` cmdlet which runs a command locally or on a remote computer (though administrator permissions are required to use PowerShell to connect to remote systems). PowerShell may also be used to download and run executables from the Internet, which can be executed from disk or in memory without touching disk. A number of PowerShell-based offensive testing tools are available, including [Empire](<https://attack.mitre.org/software/S0363>), [PowerSploit](<https://attack.mitre.org/software/S0194>), [PoshC2](<https://attack.mitre.org/software/S0378>), and PSAttack.(Citation: Github PSAttack) PowerShell commands/scripts can also be executed without directly invoking the `powershell.exe` binary through interfaces to PowerShell's underlying `System.Management.Automation` assembly DLL exposed through the .NET framework and Windows Common Language Interface (CLI).(Citation: Sixdub PowerPick Jan 2016)(Citation: SilentBreak Offensive PS Dec 2015)(Citation: Microsoft PSfromCsharp APR 2014)

**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**

T1569

**ID**

T1569

**Description**

Adversaries may abuse system services or daemons to execute commands or programs. Adversaries can execute malicious content by interacting with or creating services either locally or remotely. Many services are set to run at boot, which can aid in achieving persistence ([Create or Modify System Process](<https://attack.mitre.org/techniques/T1543>)), but adversaries can also abuse services for one-time or temporary execution.

**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**

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)  
Some files and directories may require elevated or specific user permissions to access.

# Indicator

**Name**

worldofprocure.com

**Pattern Type**

stix

**Pattern**

[domain-name:value = 'worldofprocure.com']

**Name**

stevenhead.ddns.net

**Pattern Type**

stix

**Pattern**

[hostname:value = 'stevenhead.ddns.net']

**Name**

b8f1fe93386003e82a148e0efd52759bc3be7bc7088537f6d031faec54870fb3

**Pattern Type**

stix

**Pattern**

[file:hashes:'SHA-256' =  
'b8f1fe93386003e82a148e0efd52759bc3be7bc7088537f6d031faec54870fb3']

**Name**

a562909c5c9b7b8c20484cd0822e2c379d36a34432ef11306bf1e1f28762aeb6

**Pattern Type**

stix

**Pattern**

[file:hashes:'SHA-256' =  
'a562909c5c9b7b8c20484cd0822e2c379d36a34432ef11306bf1e1f28762aeb6']

**Name**

<https://worldofprocure.com/worldofprocure.rar>

**Pattern Type**

stix

**Pattern**

[url:value = 'https://worldofprocure.com/worldofprocure.rar']

# Malware

## Name

AsyncRAT

# 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**

**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**

**Name**

**Name**

**Name**

**Name**

**Name**

# based-on

<b>Name</b>
<b>Name</b>
<b>Name</b>
<b>Name</b>
<b>Name</b>

# Domain-Name

## Value

worldofprocure.com

# StixFile

## Value

a562909c5c9b7b8c20484cd0822e2c379d36a34432ef11306bf1e1f28762aeb6

b8f1fe93386003e82a148e0efd52759bc3be7bc7088537f6d031faec54870fb3

# Hostname

## Value

stevenhead.ddns.net

# External References

- 
- <https://asec.ahnlab.com/en/67861/>
- 
- <https://otx.alienvault.com/pulse/668e52df16798200b106ada2>