Apr 15 2024

NETMANAGEIT

Intelligence Report

Automating Pikabot's String Deobfuscation

72f1a5476a845ea02344...
b178620d56a92767265...
[T1005] T1005
[T1573] T1573
1c125a10c33d862e6179...
[T1059.005] T1059.005
[T1059.001] T1059.001
Pikabot
4c53383c1088c069573f...
aebff5134e07a1586b91...
Pikabot
62f2adbc73cbdde282ae...
a9f0c978cc8519597...
e97fd71f076a7724e6...
[T1195] T1195
[T1059.003] T1059.003
15e4de42f49ea4041e40...
based on
uses
indicates

Table of contents

Overview

Entities

Observables

External References

Overview

Description

This report provides an analysis of Pikabot, a malware loader that emerged in early 2023 and employed advanced string encryption techniques to evade detection. It explains the obfuscation method used by Pikabot, which involved a combination of AES-CBC and RC4 algorithms for encrypting binary strings. The report presents an IDA plugin developed by the authors to assist in binary analysis by automating the process of decrypting Pikabot's obfuscated strings. It outlines the technical approach used in the plugin and provides the source code for the plugin.

Confidence

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

100 / 100

Content

N/A

Indicator

Name
e97fd71f076a7724e665873752c68d7a12b1b0c796bc7b9d9924ec3d49561272
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = 'e97fd71f076a7724e665873752c68d7a12b1b0c796bc7b9d9924ec3d49561272']
Name
b178620d56a927672654ce2df9ec82522a2eeb81dd3cde7e1003123e794b7116
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = 'b178620d56a927672654ce2df9ec82522a2eeb81dd3cde7e1003123e794b7116']
Name

aebff5134e07a1586b911271a49702c8623b8ac8da2c135d4d3b0145a826f507

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' =
'aebff5134e07a1586b911271a49702c8623b8ac8da2c135d4d3b0145a826f507']

Name

a9f0c978cc851959773b90d90921527dbf48977b9354b8baf024d16fc72eae01

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' =
'a9f0c978cc851959773b90d90921527dbf48977b9354b8baf024d16fc72eae01']

Name

72f1a5476a845ea02344c9b7edecfe399f64b52409229edaf856fcb9535e3242

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '72f1a5476a845ea02344c9b7edecfe399f64b52409229edaf856fcb9535e3242']
Name
62f2adbc73cbdde282ae3749aa63c2bc9c5ded8888f23160801db2db851cde8f
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '62f2adbc73cbdde282ae3749aa63c2bc9c5ded8888f23160801db2db851cde8f']
Name
4c53383c1088c069573f918c0f99fe30fa2dc9e28e800d33c4d212a5e4d36839
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '4c53383c1088c069573f918c0f99fe30fa2dc9e28e800d33c4d212a5e4d36839']
Name
1c125a10c33d862e6179b6827131e1aac587d23f1b7be0dbcb32571d70e34de4
Pattern Type

stix

Pattern

[file:hashes.'SHA-256' =
'1c125a10c33d862e6179b6827131e1aac587d23f1b7be0dbcb32571d70e34de4']

Name

15e4de42f49ea4041e4063b991ddfc6523184310f03e645c17710b370ee75347

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' =
'15e4de42f49ea4041e4063b991ddfc6523184310f03e645c17710b370ee75347']

Intrusion-Set

Name
Pikabot

Malware

Name
Pikabot

Attack-Pattern

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

T1059.003

ID

T1059.003

Description

Adversaries may abuse the Windows command shell for execution. The Windows command shell ([cmd](https://attack.mitre.org/software/S0106)) is the primary command prompt on Windows systems. The Windows command prompt can be used to control almost any aspect of a system, with various permission levels required for different subsets of commands. The command prompt can be invoked remotely via [Remote Services](https://attack.mitre.org/techniques/T1021) such as [SSH](https://attack.mitre.org/techniques/T1021/004).(Citation: SSH in Windows) Batch files (ex: .bat or .cmd) also provide the shell with a list of sequential commands to run, as well as normal scripting operations such as conditionals and loops. Common uses of batch files include long or repetitive tasks, or the need to run the same set of commands on multiple systems. Adversaries may leverage [cmd](https://attack.mitre.org/software/S0106) to execute various commands and payloads. Common uses include [cmd](https://attack.mitre.org/software/S0106) to execute a single command, or abusing [cmd](https://attack.mitre.org/software/S0106) interactively with input and output forwarded over a command and control channel.

Name

T1573

ID

T1573

Description

Adversaries may employ a known encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Despite the use of a secure algorithm, these implementations may be vulnerable to reverse engineering if secret keys are encoded and/or generated within malware samples/configuration files.

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

T1195

ID

T1195

Description

Adversaries may manipulate products or product delivery mechanisms prior to receipt by a final consumer for the purpose of data or system compromise. Supply chain compromise can take place at any stage of the supply chain including: * Manipulation of development tools * Manipulation of a development environment * Manipulation of source code repositories (public or private) * Manipulation of source code in open-source dependencies * Manipulation of software update/distribution mechanisms * Compromised/infected system images (multiple cases of removable media infected at the factory)(Citation: IBM Storwize)(Citation: Schneider Electric USB Malware) * Replacement of legitimate software with modified versions * Sales of modified/counterfeit products to legitimate distributors * Shipment interdiction While supply chain compromise can impact any component of hardware or software, adversaries looking to gain execution have often focused on malicious additions to legitimate software in software distribution or update channels.(Citation: Avast CCleaner3 2018)(Citation: Microsoft Dofoil 2018)(Citation: Command Five SK 2011) Targeting may be specific to a desired victim set or malicious software may be distributed to a broad set of consumers but only move on to additional tactics on specific victims.(Citation: Symantec Elderwood Sept 2012)(Citation: Avast CCleaner3 2018)(Citation: Command Five SK 2011) Popular open source projects that are used as dependencies in many applications may also be targeted as a means to add malicious code to users of the dependency.(Citation: Trendmicro NPM Compromise)

Name

T1005

ID

T1005

Description

Adversaries may search local system sources, such as file systems and configuration files or local databases, to find files of interest and sensitive data prior to Exfiltration. Adversaries may do this using a [Command and Scripting Interpreter](https://attack.mitre.org/techniques/T1059), such as [cmd](https://attack.mitre.org/software/S0106) as well as a [Network Device CLI](https://attack.mitre.org/techniques/T1059/008), which have functionality to interact with the file system to gather information.(Citation:

show_run_config_cmd_cisco) Adversaries may also use [Automated Collection](https://attack.mitre.org/techniques/T1119) on the local system.

StixFile

Value
e97fd71f076a7724e665873752c68d7a12b1b0c796bc7b9d9924ec3d49561272
b178620d56a927672654ce2df9ec82522a2eeb81dd3cde7e1003123e794b7116
aebff5134e07a1586b911271a49702c8623b8ac8da2c135d4d3b0145a826f507
a9f0c978cc851959773b90d90921527dbf48977b9354b8baf024d16fc72eae01
72f1a5476a845ea02344c9b7edecfe399f64b52409229edaf856fcb9535e3242
62f2adbc73cbdde282ae3749aa63c2bc9c5ded8888f23160801db2db851cde8f
4c53383c1088c069573f918c0f99fe30fa2dc9e28e800d33c4d212a5e4d36839
1c125a10c33d862e6179b6827131e1aac587d23f1b7be0dbcb32571d70e34de4
15e4de42f49ea4041e4063b991ddfc6523184310f03e645c17710b370ee75347

External References

- https://www.zscaler.com/blogs/security-research/automating-pikabot-s-string-deobfuscation
- https://otx.alienvault.com/pulse/661ce4d7a2518a36e402343f