Financial Institutions in the Sight of New JsOutProx Attack Waves

yoroi.company/research/financial-institutions-in-the-sight-of-new-jsoutprox-attack-waves/

August 31, 2021

08/31/2021

Introduction

When threat actors evolve, their tools do so. Observing the evolution of the threats we track during our cyber defense operations is part of what we do to secure our customers. Back in 2019, the Yoroi's Malware ZLAB unit discovered a complete new malware implant named "JsOutProx" (TH-264), a complex JavaScript-based RAT used to attack financial institutions in the APAC area.

In the last two years, the evolution of this implant was clear. After our initial discovery, many security research teams started monitoring this elusive threat, both ZScaler researchers and Fortinet ones began tracking the actor activities too, then we started noticing the sophistication of the malicious code. The actor behind JsOutProx (TH-264), also dubbed as SOLAR SPIDER in the CrowdStrike naming scheme, is protecting the new code with improved protection mechanisms and at the same time we started noticing signs of a potential emergent expansion of the attack operations to western financial organizations.

In this report, Yoroi's Malware ZLAB dissects the latest version of JsOutProx malware and defeats its new protection mechanisms to unveil its full offensive capabilities.

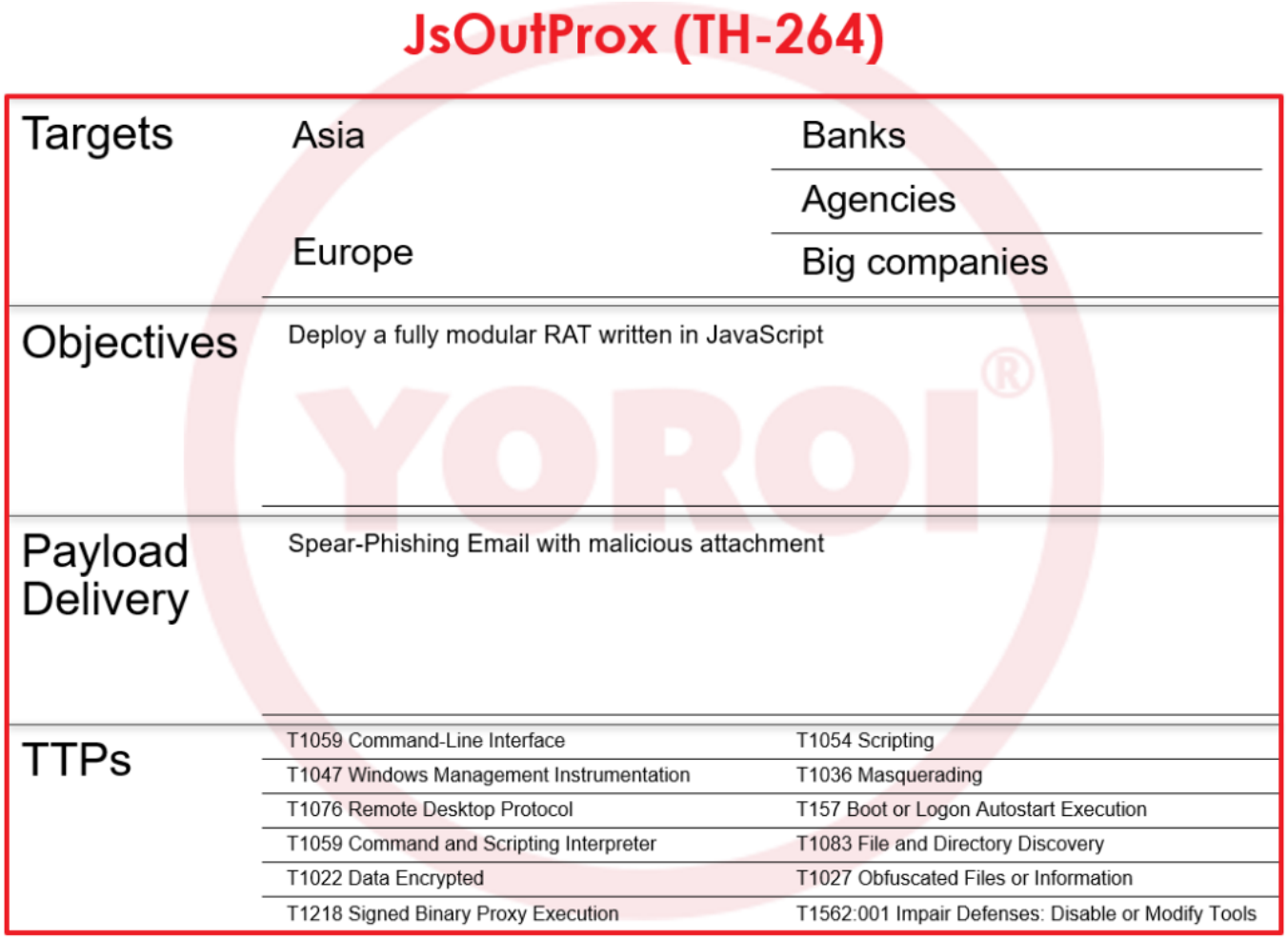

Figure 1: Yoroi Flashcard about JsOutProx

Technical Analysis

Hash	65987f95b365501579431ea8dec1d45940430d8c9defad58908a14e6fb96a347
Threat	JsOutProx
Brief Description	JsOutProx RAT – Jun 2021
File Size	2276KB
Ssdeep	24576:eOa0QS/9pn7qKkxds34NPoE3j2hYW8jecNH2Pzw83ZEkTE50DngGfayxg3qfDOoX:Pl

Table 1: Static information about the sample

The infection starts with a malicious phishing messages attending to be a bank transaction. It also leverages the classic Masquerading technique (T1036), pretending to be a PDF file instead of a JS code.

Besides this elementary technique, we decided to deepen the improvements of the various versions we observed. The first immediate basic static evidence which we noticed is the increasing of the file size. In fact, we started to analyze the first documented version which was about 800KB, and now it is three time the original one, letting immediately think about the fact that the malware writers improved the obfuscation strategies, and not only, they enriched the capabilities of the malicious code.

Digging inside the new JsOutProx version

This new variant has a more sophisticated level of obfuscation than the previous versions, enhancing the anti-analysis capabilities. The high-level structure of the code can be synthetized in the following figure:

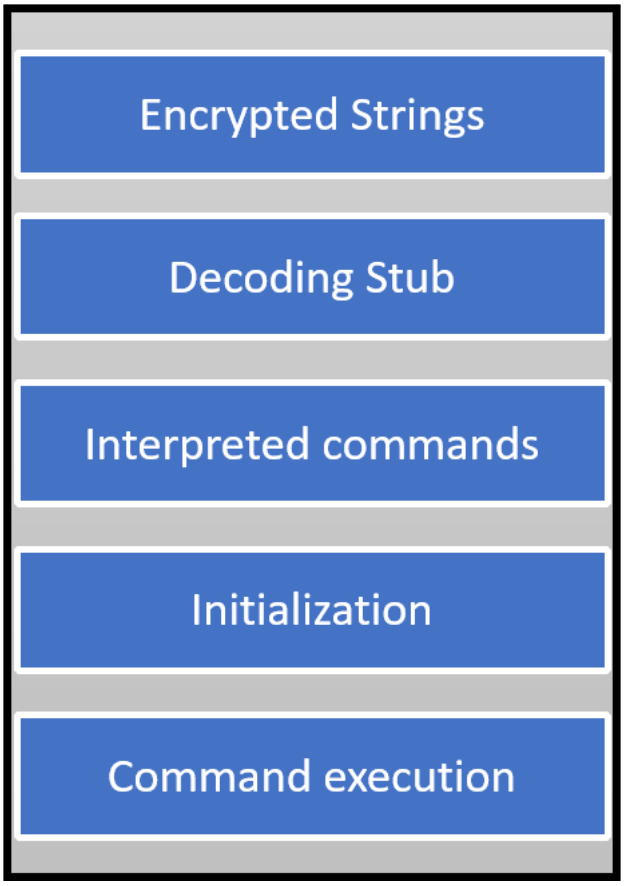

Figure 2: High level structure of the malicious JavaScript

The first part of the malicious code is a long array containing about 30k of encrypted strings encoded in Base-64 format:

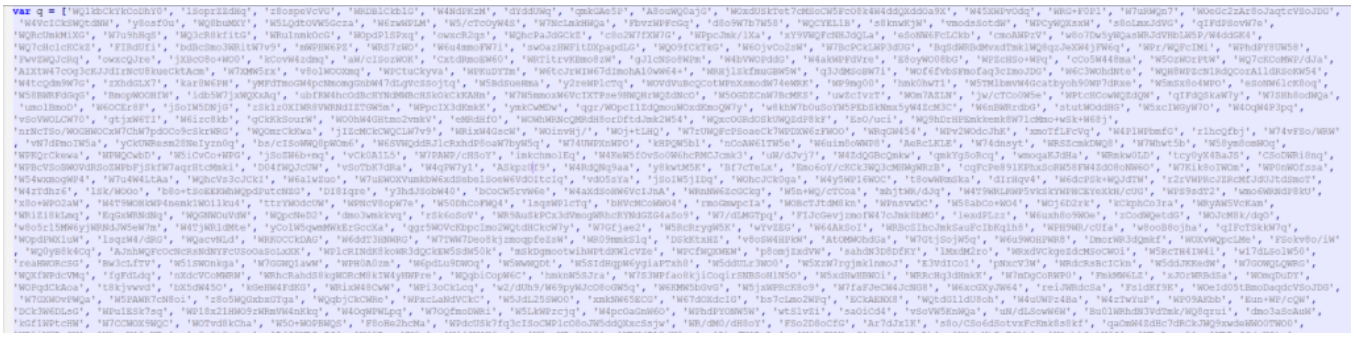
Figure 3: Piece of the long array containing the encrypted information

After that, we spotted a piece of code looking as a decoding stub, composed by the following primitives:

```
var uc = function(h, L, Z, R, y) {
var uD = function(h, L, Z, R, y) {
var uE = function(h, L, Z, R, y) {
var uv = function(h, L, Z, R, y) {
var uT = function(h, L, Z, R, y) {
(function(h, L) {
 var uw = [uc('0x714b', '7FFQ', '0x544f', '0x5491', '0x7eea') + uD('0x2da7', 'tc
(function(L, Z) {
var uN = function(L, Z) {
function H(h, L) {
```
Figure 4: Prototipe of the decoding routines

The most interesting function of the ones reported below is the “H” function because it hides the main decoding routine.

Figure 5: Piexe of the recursive decoding routine

This function represents unambiguous evidence of how the attacker enhanced the code obfuscation because the body has been rewritten to be recursive, making the analysis harder and to avoid detection.

After that decoding stub, we isolated the configuration of the malware: this time the structure of the configuration is similar to the previous versions of JsOutProx. After the deobfuscation operations we obtained an acceptable human-readable result:

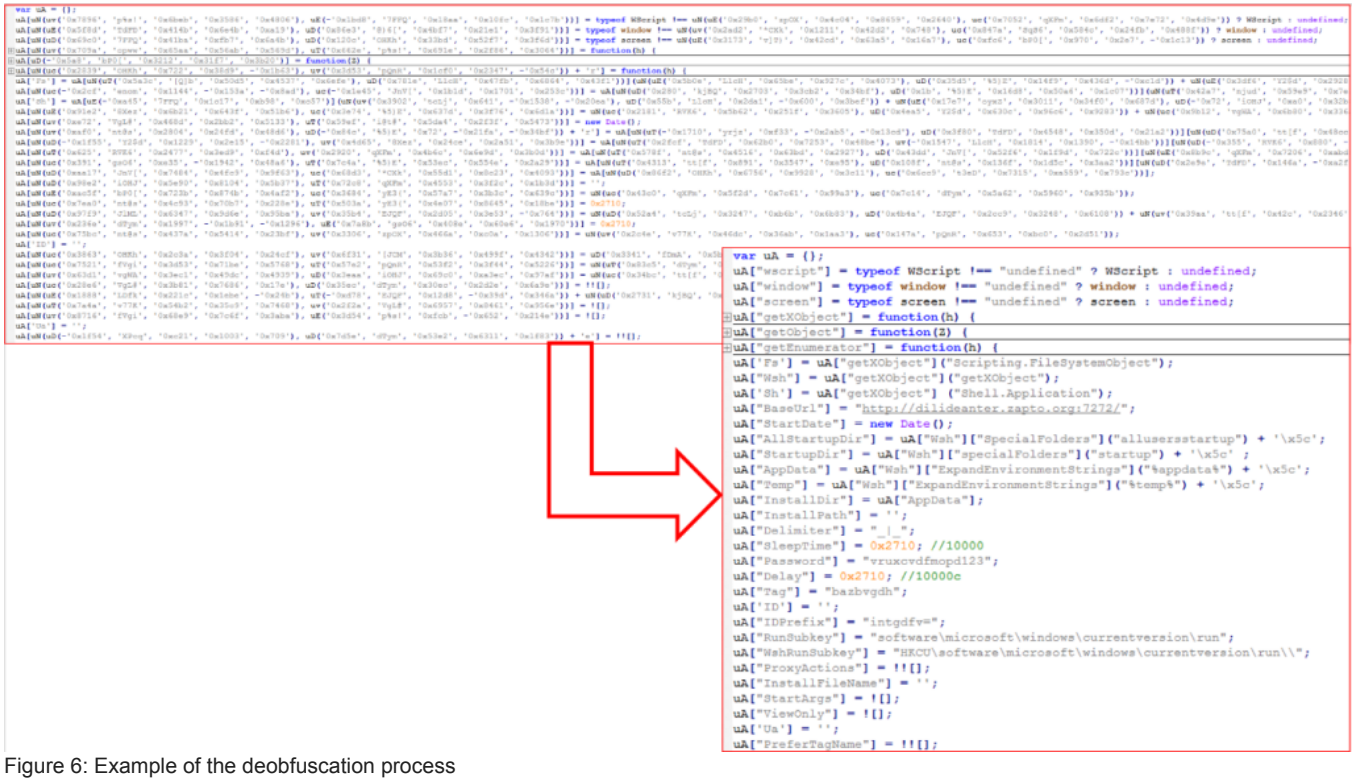

Figure 6: Example of the deobfuscation process

The configuration of the malware is substantially similar since le initial versions of the samples but there are a couple field we decided to report the most relevant of them:

Field name	Description
Fs	Gain FileSystem privileges to Read-Write files
Wsh	Allow the program to execute WScript files
Sh	Allow the program to execute command line programs
BaseUrl	C2 URL
StartDate	Date of the starting infection
InstallDir	Directory of the malware installation
Delimiter	Separator used by the malware to delimit the exfiltrated information
SleepTime	Delay of the malware Execution
Password	A password probably used to decrypt some content in some
Tag	Tag of the campaign (which we initially used to name the malware family as JsOutProx)
IDPrefix	A parameter set during the sending the Cookie in requests to the C2
ProxyActions	*A new parameter indicating if the malware is a "proxy" for other process. In particular, the malware starts every time it needs to execute a command a new process from the command line*
StartArgs	A series of arguments allowing the malware to customize the infection
ViewOnly	*A new boolean parameter indicating whether the infection is in view only mode: if it is so, the malware cannot modify anything on the system, but only exfiltrate and modify the malware*

Table 2: Configuration of JsOutProx

Two of these configuratons are particularly interesting to understand the flexibility of the implant: "*ProxyActions*" and "*ViewOnly*". This malware can also operate as a silent info stealer making it an impressive reconnaissance tool for the initial phases of an intrusion, but it can also be leveraged to run more advanced, and noisy, offensive plugins using proxy processes to keep the main infection process safe from detection and security terminations.

Offensive Capabilities

After that, we have an exceptionally extensive list of interpreted commands used by the malware. Over the years, these commands have been modified and their body changed, so we decided to report a comprehensive list of them to provide a clear outlook about the offensive capabilities of this javascript-based malware:

Interpreted Commands	Description
["getUa"]	Ua stands for the interpreter of that commands
["getUuid"]	Retrieves the Uuid of the malicious process
["isHTA"]	Return true if the malicious script is an HTA file
["isWScript"]	Return true if the malicious script is a WScript-readable file
["sleepPing"]	Use ping to localhost as a sleep technique
["sleep"]	Sleep for a set time
["sleepEx"]	Sleep for a set time
["scriptName"]	Retrieve the script name
["scriptFullName"]	Retrieve the full name of the malicious script
["scriptParentDir"]	Retrieve the path of the script
["launch"]	Launch a command
["getJavaPath"]	Retrieve the java path
["checkJrePath"]	Check the existence of JRE

["getLaunchCommand"]	Retrieve the command
["installOrRun"]	Decide if install or run without persistence
["install"]	Install the malware persistence
["unInstall"]	Uninstall the malware
["getLauncher"]	Get the Launcher Process (it can be MSHTA, WScript, Powershell)
["addRegistry"]	Write a new Registry key in RunSubKey config key
["removeRegistry"]	Remove a new Registry key in RunSubKey config key
["addStartup"]	Set the persistence through the the Autorun Registry Key
["removeStartup"]	Remove the persistence
["exit"]	Kill the malware process
["getFullBody"]	Retrieve the full body of the response of the C2 with all the commands separated by the Delimiter
["setZone"]	Set a specific Time zone
["update"]	Update the malware Core
["restart"]	Restart the malware process
["receive"]	Receive is a listener to interpret the commands sent by the C2. The commands are: “fnm”: send ScriptFullName to the invoke DownloadPlugin “sp”: invoke ScreenPShellPlugin “cn”: invoke ShellPlugin “in”: invoke InfoPlugin “sh”: invoke S
["launchArgs"]	Execute the malware with the arguments
["getArgs"]	Retrieve the arguments of the malware
["init"]	Malware initializations
["Base64"]	A sort of constructor for Base64 commands family
["Base64"]["encode"]	Encode a stream in Base64 format
["Base64"] ["encodeString"]	Encode a string in Base64 format
["Base64"]["encodeFile"]	Encode a file in Base64 Format
["Base64"]["decode"]	Decode a file or a stream from Base64Format
["Base64"] ["decodeString"]	Decode a string from Base64Format
["Environment"]	A constructor for Environment commands
["Environment"] ["userName"]	Retrieve the username of the infected machine
["Environment"] ["computerName"]	Retrieve the computer name of the infected machine
["Environment"] ["userDomain"]	Retrieve the username inside the domain
["Environment"] ["encoding"]	Retrieve the encoding of the strings by reading the registry key: “HKLM\SYSTEM\CurrentControlSet\Control\Nls\Code
["Environment"] ["engineVersion"]	Retrieve the engine version of the interpreter
["Environment"] ["engine"]	Set a specific engine interpreter
["Hex"]	A sort of connstructor for the Hex command family
["Hex"]["encodeBytes"]	Encode a byte-stream in hex format
["Hex"]["encode"]	Encode a stream in hex format
["Hex"]["encodeFile"]	Encode a file in Hex Format

["Hex"]["decodeBytes"]	Decode a stream from the Hex Format
["Hex"]["decode"]	Decode a stream or a file from the hex format
["Http"]	A sort of constructor oh Http Command family
["Http"]["create"]	Construct a "WinHttp.WinHttpRequest.5.1" request
["Http"]["addHeaders"]	Add new Headers to the HttpRequest
["Http"]["post"]	Prepare the Post Request
["Http"]["get"]	Prepare the Get Request
["Http"]["head"]	Prepare the Head Request
["Http"]["send"]	Send efiltrated info to the C2
["Http"]["uploadFile"]	Upload a File to the C2
["Http"]["downloadFile"]	Download a file from the C2
["MessageBag"]	A sort of constructor for internal log messages
["MessageBag"]["info"]	Log an info
["MessageBag"]["error"]	Log an error
["Moment"]	A sort of constructor for Moment object
["Moment"] ["getDayString"]	Retrieve the exact time of the machine
['Os']	A sort od constructor for Os commands
['Os']["caption"]	Get a screenshot of the machine
['Os']["security"]	Retrieve che security configuration through WMI query
['Os']["isResolution"]	Retrieve the Resolution of the HTA application of the malware
['Os'] ["screenResolution"]	Retrieve the screen resolution
['Os'] ["wmiScreenResolution"]	Retrieve the screen resolution through WMI command
['Os'] ["wmiMonitorCount"]	Retrieve the number of monitors through WMi command
['Os']["arch"]	Retrieve the CPU architecture info
['Os']["volumeSerial"]	Retrieve the Serial of the disk
['Os']["version"]	Retrieve the OS version through winmgmts command
['Os']["version2"]	Retrieve the OS version using the ['Os']["version"] primitive
['Os']["windowsXp"]	Check is WindowsXP
['Os']["majorVersion"]	Retrieve only the major version
["Registry"]	A sort of constructor of Registry commands
["Registry"]["HKCR"] = 0x80000000	Set a sort of instance variable for HKCR
["Registry"]["HKCU"] = 0x80000001	Set a sort of instance variable for HKCU
["Registry"]["HKLM"] = 0x80000002	Set a sort of instance variable for HKLM
["Registry"]["HKUS"] = 0x80000003	Set a sort of instance variable for HKUS
["Registry"]["HKCC"] = 0x80000005	Set a sort of instance variable for HKCC

["Registry"]["STRING"] = 0x0	Set a sort of instance variable for String Variable
["Registry"]["BINARY"] = 0x1	Set a sort of instance variable for Binary Variable
["Registry"]["DWORD"] = 0x2	Set a sort of instance variable for DWORD variable
["Registry"]["QWORD"] = 0x3	Set a sort of instance variable for QWORD Varibale
["Registry"] ["WOW64_32KEY"] = 0x20	Set a sort of instance variable for 32 bit key
["Registry"] ["WOW64_64KEY"] = 0x40	Set a sort of instance variable for 64 bit key
["Registry"]["provider"]	Get the provider to access to Registry
["Registry"]["write"]	Wrtie a Registry key
["Registry"]["keyExists"]	Check if a registry key exists
["Registry"]["read"]	Read a Registry key
["Registry"]["remove"]	Remove a registry key
["Registry"] ["removeValue"]	Remove the value of a registry key
["Registry"]["create"]	Create a registry key
["Stream"]	A sort of Constructor for Stream Utility
["Stream"] ["stringToBytes"]	Convert a string to a byte array stream
["Stream"] ["bytesToString"]	Convert a byte array string to a string
["XmlEncoding"]	A sort of Constructor for XmlEncoding Utility
["XmlEncoding"] ["encodeString"]	Encode a string in Microsoft.XMLDOM format using ["XmlEncoding"]["encode"] commadn
["XmlEncoding"] ["encode"]	Encode a Stream in Microsoft.XMLDOM format
["XmlEncoding"] ["encodeFile"]	Encode a file in Microsoft.XMLDOM format
["XmlEncoding"] ["decode"]	Decode a Stream encoded in Microsoft.XMLDOM format
["XmlEncoding"] ["decodeString"]	Decode a String encoded in Microsoft.XLDOM format
["Association"]	A sort of constructor for Association commands
["Association"] ["createAssocCmd"]	Prepare a Command line to execute through [“Process”] commands
["Association"] ["createAssoc"]	Prepare a command line string to pass to methods
["Association"] ["createTypeCmd"]	Create a new process command line
["Association"] ["createType"]	Create a command line to exec by writing a command inside "HKCR\\{0}\\shell\\open\\command\\" registry key
["Association"] ["removeAssocCmd"]	Remove an associaction through another command line
["Association"] ["removeTypeCmd"]	Remove an association through the registry key

["Association"] ["removeAssoc"]	Remove the prepared command line
["Association"] ["removeType"]	Remove the prepared command ilne to pass to the registry key
["AssociationPlugin"]	A sort of constructor for the AssociationPlugin
["AssociationPlugin"] ["receive"]	A listener for the Commands to exec to Association. The commands are: "as.g": Send to the C2 through [Http][Send] t
["Download"]	A sort of constructor for Download comamnd
["Download"]["pShell"]	Download a file through a Powershell command
["DownloadPlugin"]	A sort of constructor for Download Plugin
["DownloadPlugin"] ["receive"]	A listener for the DownloadPlugin module. The command is named “do.n”
["File"]	A sort of constructor for File commands
["File"]["waitFor"]	Wait for a specific Date
["File"]["getParentPath"]	Retrieve the parent Path
["File"]["getFileName"]	Retrieve the File name of the malware
["File"]["remove"]	Remove a file
["File"]["copy"]	Copy a File
["File"]["move"]	Move a File
["File"]["createFolder"]	Create a Folder
["File"]["removeEx"]	Remove a File or directory through cmd line
["File"]["copyEx"]	Copy a file or directory through cmd line
["File"]["moveEx"]	Move a file or directory through cmd line
["File"]["rename"]	Rename a file or directory
["File"]["createFolderEx"]	Create a folder though cmd line
["File"]["getInfo"]	Retrieve the info of a folder or a file
["File"]["getSize"]	Retrieve the size of a folder or a file
["File"]["exists"]	Check if a file or a folder exists
["File"]["fileExists"]	Check if a file exists
["File"]["folderExists"]	Check if a folder exists
["File"]["driveExists"]	Check if a drive Exists
["File"]["expandPath"]	Expand the Environmental strings
["File"]["getFileSize"]	Retrieve a file size
["File"]["getFolderSize"]	Retrieve a Folder Size
["File"]["deleteFile"]	Delete a file
["File"]["deleteFolder"]	Delete a Folder
["File"]["copyFile"]	Copy a File
["File"]["moveFile"]	Move a File
["File"]["copyFolder"]	Copy a folder
["File"]["moveFolder"]	Move a Folder
["File"]["getFileInfo"]	Retrieve the information of a file
["File"]["getFolderInfo"]	Retrieve the folder info

["File"]["unZip"]	Unzip all zip file inside a folder
["File"]["createZip"]	Create a zip
["File"]["zip"]	Zip a file
["File"]["writeAllText"]	Write a Text through getXObject
["File"]["readAllBytes"]	Read a binary file through getXObjext
["File"]["writeAllBytes"]	Write all bytes
["File"]["dumpDrives"]	Enumerate all the drives drives
["File"]["dumpPath"]	Enumerate all the files inside a folder
["File"]["spawn"]	Execute a new File (exe or script)
["File"]["getPreview"]	Get a preview of an image File through a powershell script
["FilePlugin"]	A constructor for FilePlugin
["FilePlugin"]["receive"]	A listener of the commands of file. The commands are: “fi.dv” Send to the c2 the dumpfiles “fi.g” Send to the C2 the fo Invoke addRegistry command
["InfoPlugin"]	A constructor for InfoPlugin command
["InfoPlugin"]["receive"]	A listener of the unique command “in.g”, where the malware sends to the C2 a long series of information about the vic
["JavaInstall"]	A constructor for JavaInstall command
["JavaInstall"]["install"]	Install JRE by downloading it from the C2
["JavaInstall"] ["unInstall"]	Uninstall Java
["JavaInstallPlugin"]	A constructor for JavaInstallPlugin
["JavaInstallPlugin"] ["receive"]	A listener to install or check the installation of JRE through the commands “jv.i” and “jv.st”
["MultiView"]	A constructor of Multiview module commands
["MultiView"] ["getCaption"]	Retrieve a ForegroundWindows through a custom PowerShell script
["MultiViewPlugin"]	A constructor for the MultiViewPlugin module commands
["MultiViewPlugin"] ["Quality"] = 0x64	A variable indicating the quality of the foregound window
["MultiViewPlugin"] ["Scale"] = 0xa	A variable indicating the scaling of the foreground window
["MultiViewPlugin"] ["receive"]	A listener to the commands of MultiView Plugin which are: “mv.st”: set Quality and Scale “mv.s”: capture the screen th
["Process"]	A constructor for Process module utility
["Process"]["list"]	List all the active processes
["Process"]["dump"]	Retrieve the following information from all processes: Caption CommandLine CreationClassName CreationDate CSCreationClassName CSName Description E
["Process"]["taskkillId"]	Kill a process by ID through taskkill
["Process"] ["taskkillName"]	Kill a process by Name through taskkill
["Process"] ["terminateId"]	Kill a process by ID through Process Terminate
["Process"] ["terminateName"]	Kill a process by name through Process Terminate
["Process"]["createWmi"]	Create a Wmi provider through WbemScripting.SWbemLocator
["Process"] ["createProcess"]	Create a process using "winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2"

["Process"]["getProcessByID"]	Retrieve a pointer to a process by ID
["Process"]["getProcessByName"]	Retrieve a pointer to a process by name
["Process"]["currentPID"]	Retrieve the current ID
["Process"]["currentPIDPS"]	Retrieve the current ID through a PowerShell
["ProcessPlugin"]	A constructor for ProcessPlugin module
["ProcessPlugin"]["receive"]	A listener for the following commands: “pr.g”: send to the C2 the result of the operation of Process dump “pr.kli”: Invok
["ScreenPShell"]	A constructor for ScreenPShell module
["ScreenPShell"]["capture"]	Get a screenshot through a PowerShell script
["ScreenPShell"]["pressKeyCode"]	Simulate the pression of a keystroke
["ScreenPShell"]["pressKeyChar"]	Simulate the pression of a keystroke
["ScreenPShell"]["sendkeys"]	Invoke Wsh sendKeys
["ScreenPShell"]["leftDown"]	Simulate the navigation
["ScreenPShell"]["leftUp"]	Simulate the navigation
["ScreenPShell"]["leftClick"]	Simulate the navigation
["ScreenPShell"]["doubleClick"]	Simulate the navigation
["ScreenPShell"]["rightDown"]	Simulate the navigation
["ScreenPShell"]["rightUp"]	Simulate the navigation
["ScreenPShell"]["rightClick"]	Simulate the navigation
["ScreenPShell"]["mouseWheel"]	Simulate the navigation
["ScreenPShell"]["scrollUp"]	Simulate the navigation
["ScreenPShell"]["scrollDown"]	Simulate the navigation
["ScreenPShell"]["move"]	Simulate the navigation
["ScreenPShell"]["getAction"]	Simulate the navigation
["ScreenPShell"]["clickAction"]	Simulate the navigation
["ScreenPShellPlugin"]	A constructor for ScreenPShellPlugin
["ScreenPShellPlugin"]["OldSize"] = 0x0	A variable for OldSize property
["ScreenPShellPlugin"]["Quality"] = 0xf	A variable for Quality property

["ScreenPShellPlugin"] ["ScreenNumber"] = -0x1	A variable for ScreenNumber property
["ScreenPShellPlugin"] ["CaptureMouse"] = !![]	A variable for CaptureMouse property
["ScreenPShellPlugin"] ["receive"]	A listener for the ScreenPShell for the following commands: “sp.set”: invoke capture mouse “sp.g”: send to C2 the ca
["Shell"] = {}	A constructor for Shell command module
["Shell"]["codePage"]	Retrieve the codePage by reading the registry key "HKLM\SYSTEM\CurrentControlSet\Control\NIs\CodePage\OEMC
["Shell"]["run"]	Execute a command through Wsh
["Shell"]["shellExecute"]	Execute a command through ShellExecute
["Shell"]["getOutput"]	Read a log file of a shell command
["ShellPlugin"]	A constructor for ShellPlugin
["ShellPlugin"]["receive"]	A listener for ShellPlugin for the following command: “cn.e”: send to the C2 the get getOutput
["Shortcut"]	A constructor for Shortcut command
["Shortcut"]["create"]	Create a Shortcut
["Shortcut"]["dump"]	Extract the information of a shortcut
["Shortcut"]["getTarget"]	Retrieve the Target of a Shortcut
["Shortcut"]["run"]	Execute a shortcut file
["ShortcutPlugin"]	A constructor for Shortcut plugin
["ShortcutPlugin"] ["receive"]	A listener for ShortcutPlugin excuting the following command: "sh.g": send to the C2 the info retrieved from Shortcut D

Table 3: Interpreted Commands list

The Initialization

At this point, we talk about the initialization of the malware. Inspecting the code after the decoding routine we noticed a simple if-else check to decide how to execute the malicious code.

The code structure is essential but provides confirmation about the preferred initial vectors to deliver the malicious implant. The Javascript malware is designed to be interpreted by the "WScript.exe" windows interpreter and through the "MSHTA" utility, a widely abused LoLBins (T1218).

```
try {
    if(uA["isWScript"]()) {
        uA["init"]();
    } else if(uA["isHTA"]())
        uA["init"]();
    } else {
        uA["init"]();
    }
} catch(um) {}
```

Figure 7: Init routine

Conclusion

The JsOutProx malware is an example of how threat actor abuses high-level scripting language to realize fully functional implants, easily evading traditional signature-based detection tools that are typically good at spotting binaries, not text.

The implant is increasing its sophisticated for three reasons: the obfuscation is becoming increasingly intense and can avoid the initial detection, the newer versions dropped the in-memory .NET modules and adopted a proxy process plugin architecture to enhance the survival of the main infection routine, also, the introduction of the “view-only mode” represents a notable change in the flexibility of this malicious tool that could be also configured to have the lowest footprint possible while keeping eyes on the victim’s desktop.

Silent attacks such as this one are hard to spot with traditional security. Yoroi protects its customers from this threat leveraging its proprietary automated malware sandbox technology, Yomi sandbox, to analyze and detect any sign of infection, even the weakest.

Indicators of Compromise

Hash:

65987f95b365501579431ea8dec1d45940430d8c9defad58908a14e6fb96a347

C2:

- dilideanter.]zapto.]org
- hxxp://dilideanter.]zapto.]org[:7272/

Yara Rules

```
rule JsOutProx_v2 {


meta:
      description = "Yara Rule for JsOutProx_v2"
      author = "Yoroi Malware Zlab"
      last_updated = "2021_07_29"
      tlp = "white"
      category = "informational"


strings:
    $s1= /uA\[[a-zA-Z]/ ascii wide
    $s2= /u[A-Z]\(/ ascii wide


condition:
    #s1>800 and #s2>4000 and (filesize > 1500KB)

}
```

This blog post was authored by Luigi Martire and Luca Mella of Yoroi Malware ZLAB.