Malware Analysis Spotlight: XLoader’ Cross-platform Support Utilizing XBinder

vmray.com/cyber-security-blog/malware-analysis-spotlight-xbinder-xloader/

XLoader’ Cross-platform Support Utilizing XBinder

From the VMRay Labs Team

Introduction

Lately, a rebranded version of the stealer FormBook named XLoader has emerged. In contrast to FormBook, which targets Windows only, XLoader supports macOS as well. During our research, we observed Office documents, which exploit vulnerabilities in MS Office products, and malicious loaders like Smoke Loader or GuLoader distributing FormBook and XLoader Windows samples.

In addition to XLoader, the developers published another tool called XBinder written in Java. It combines samples that target different operating systems into one single file. Given the executable file format differs between Windows and macOS, it is most likely the purpose of this tool to ease the platform-independent distribution of samples.

In this Spotlight, we take a closer look at XBinder and XLoader's behavior. We adopt the perspective of an attacker and use XBinder to combine two public XLoader samples into one single sample that we are going to analyze further.

View the VMRay Analyzer Report for XLoader macOS and XLoader Windows

Distribution with XBinder

As mentioned before, XBinder is written in Java and combines two samples into one a Java Archive sample. Therefore, the generated sample is executable on both operating systems as long as the Java Runtime Environment (JRE) is installed. In case the JRE is missing, the Java Archive can't be executed. Besides the samples for the operating systems Windows and macOS, XBinder requests a display file that it shows to the victim at execution (Figure 1). Attackers can use this display file to remain inconspicuous by, for example, executing benign software or showing an error message. For this Spotlight, we have chosen a simple text file as the display file.

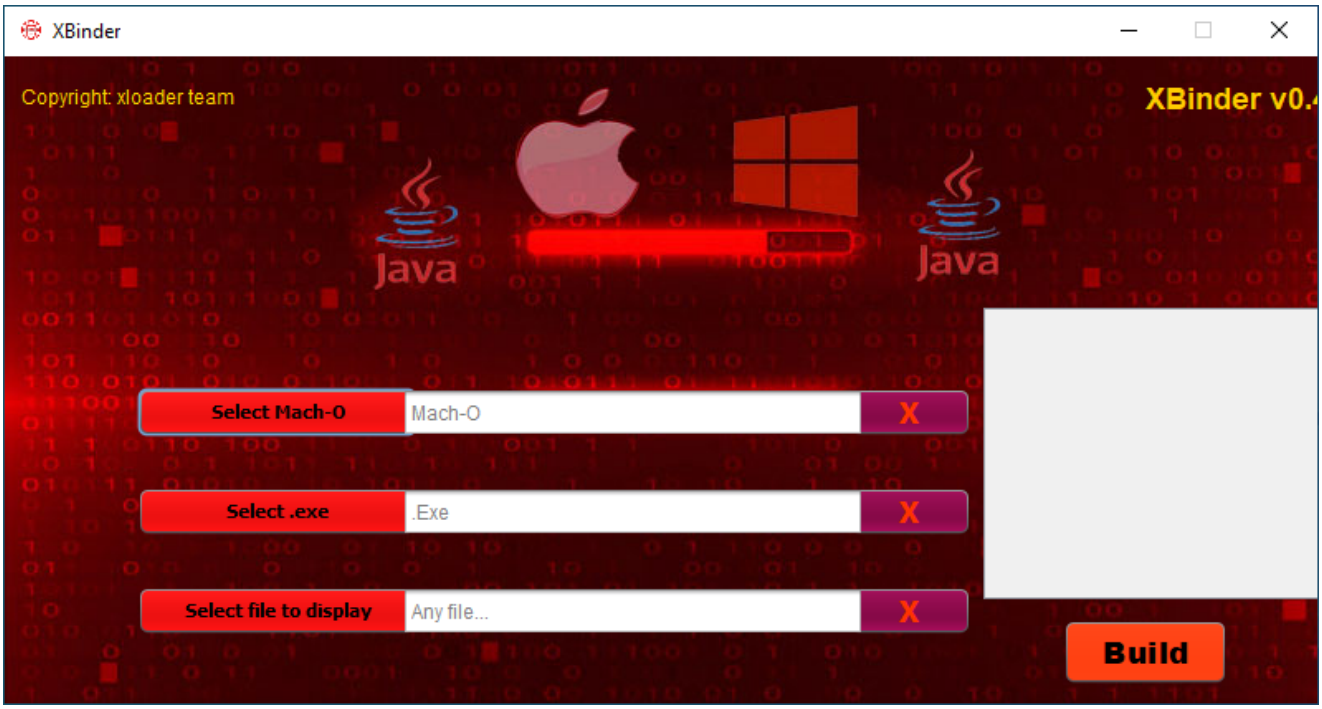


Figure 1: XBinder Builder

By submitting the generated sample to the VMRay Platform and selecting a macOS and Windows Virtual Machine, we can observe its behavior and the behavior of both XLoader payloads (see XLoader macOS and XLoader Windows). We can see that in both analyses, the sample tries to evade a debugger, achieves persistence, and access system data (Figure 2).

Figure 2: VMRay Platform – VTI indicating XLoader and XBinder on Windows (left) and macOS (right).

Figure 3. shows the process graphs generated by the VMRay Platform for Windows (top) and macOS (bottom) of the generated sample. In both analyses, the sample leads to payload execution and showing the display file using TextEdit or Notepad.

The payloads are dropped in the user's home directory (%USERPROFILE% or $HOME). These dropped files are the respective XLoader payloads selected above.

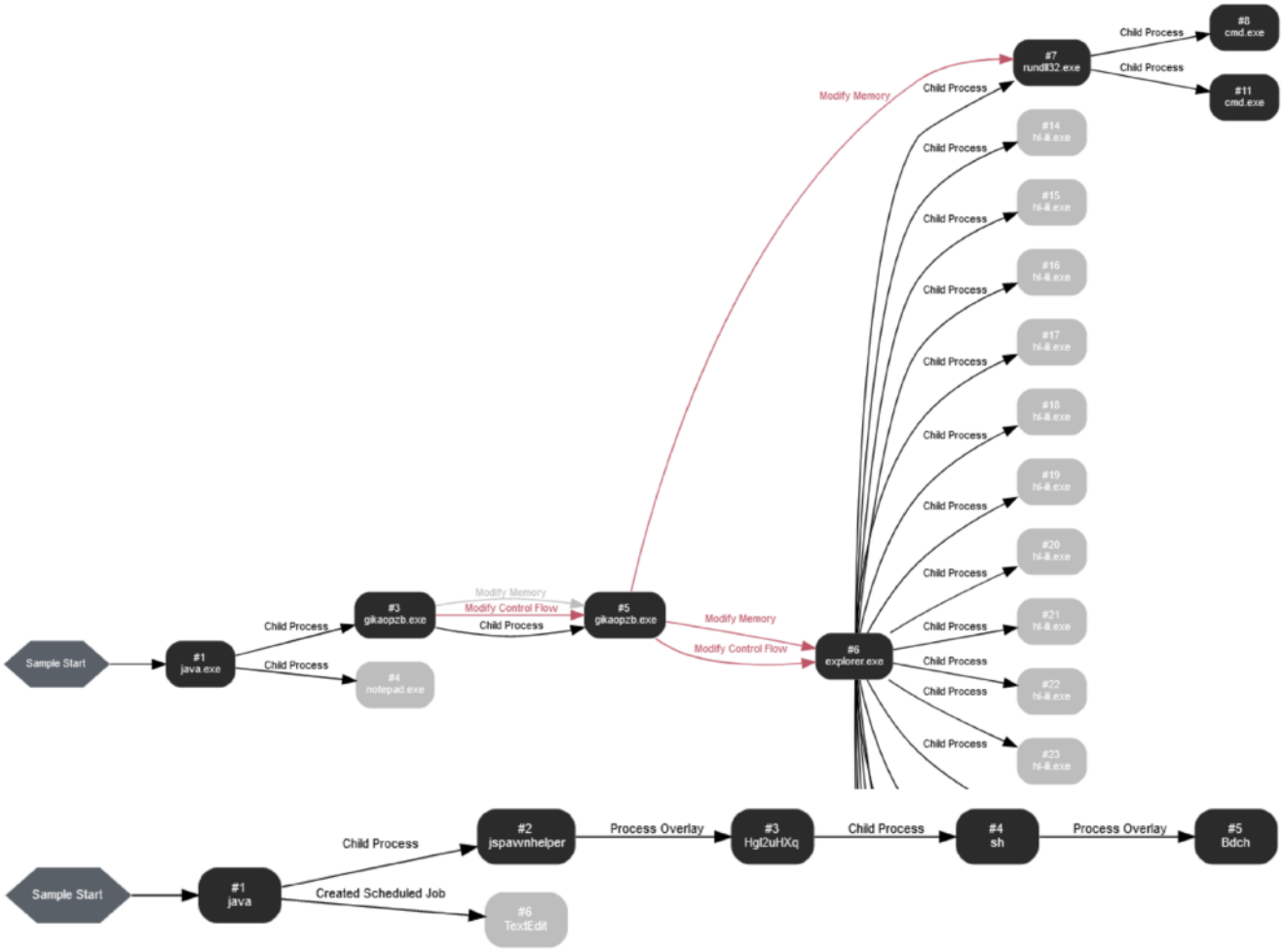


Figure 3: VMRay Platform – Process graph of both analyses

Figure 4. shows the main function of the decompiled sample file and its content. Both payloads and the display file are encrypted with the AES (Advanced Encryption Standard) and stored as three separate resources. Upon execution, the sample determines the current operating system, decrypts the display file and respective payload, and performs the previously seen behavior.

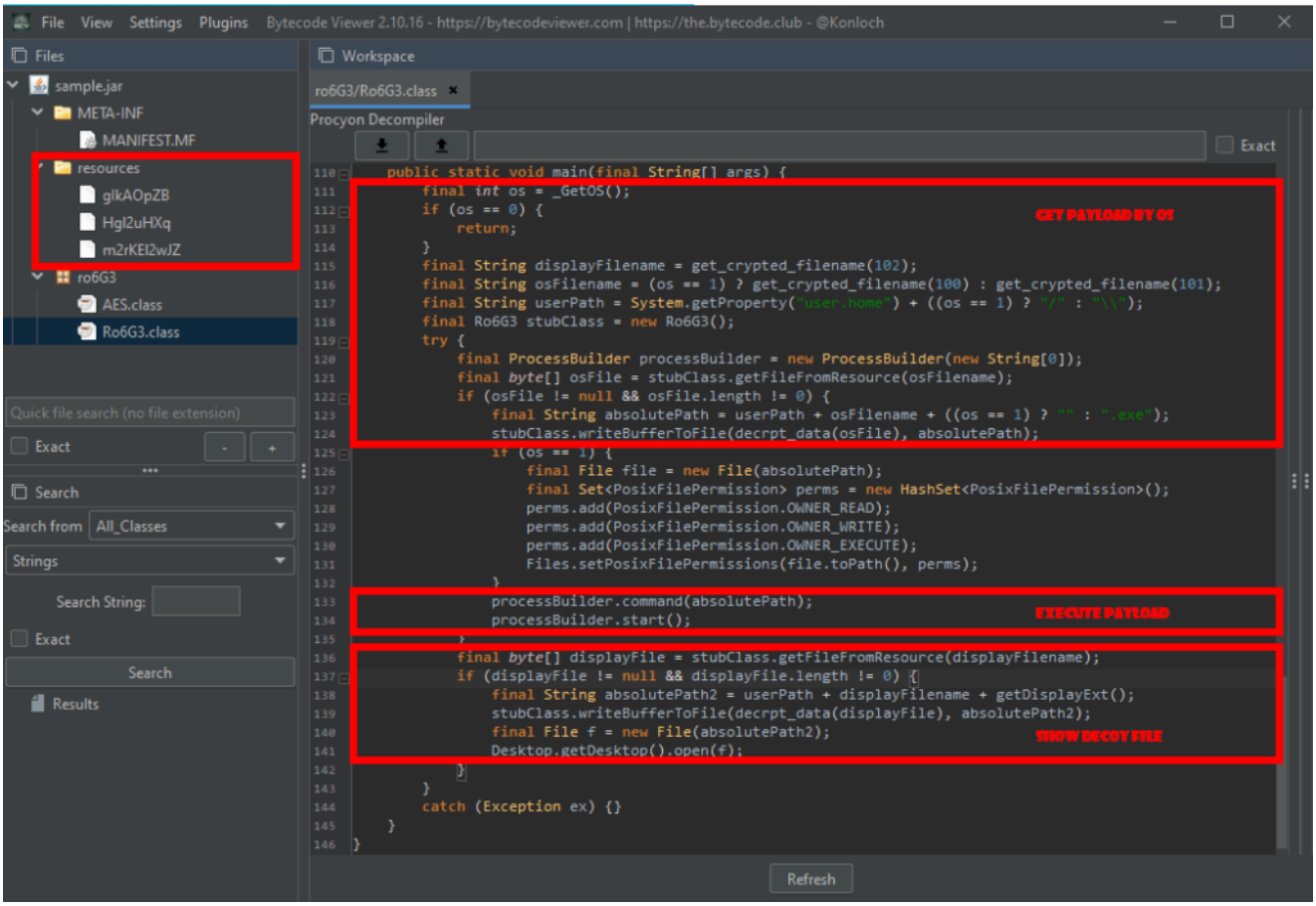


Figure 4: Decompiled XBinder sample

So far, we have seen that three different (executable) file types are involved in the analyses. Because XLoader is compiled natively, it can only be executed on either Windows or macOS but not on both. To achieve cross-platform support at the sample level, actors can use XBinder and benefit from Java's cross-platform support.

Comparison of XLoader Payloads

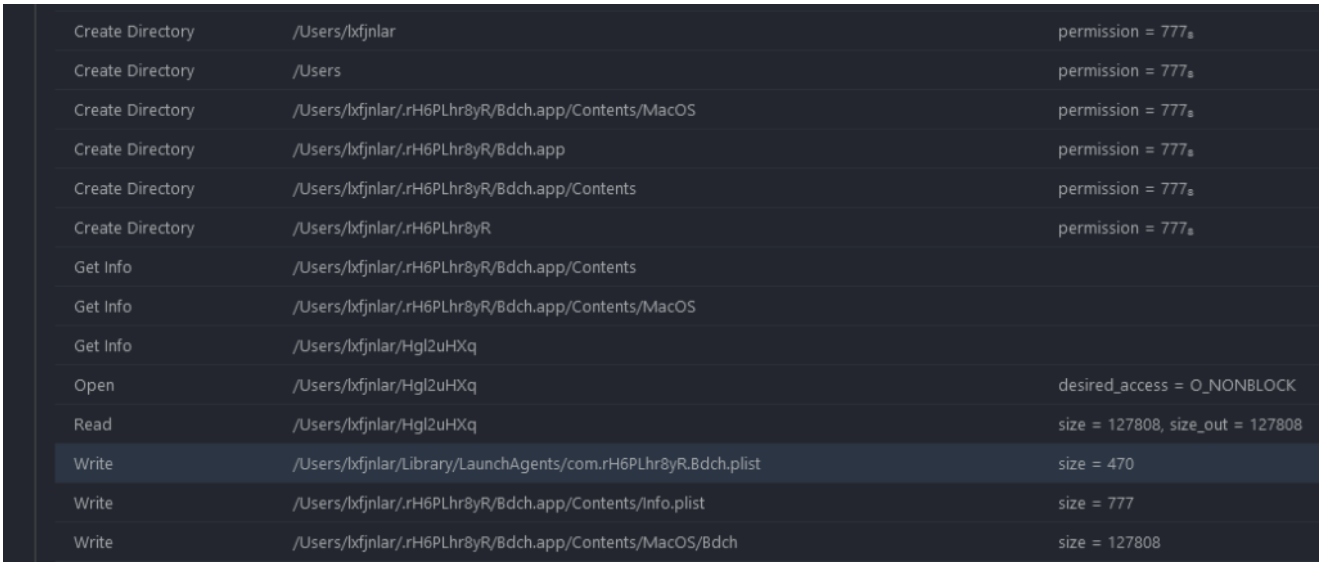
Up until now, we have seen how the generated example executes the payloads and how XBinder adds cross-platform support on the sample level. Next, lets take a look at the behavior of both payloads.

At first glance, the behavior of the Windows payload is similar to FormBook's. For example, both abuse the explorer process to spawn a new process of a legitimate Windows binary. Next, they inject into this created process which has explorer as its parent and

therefore looks like it was initially started by the user. FormBook and XLoader Windows steal credentials from browsers including Mozilla Firefox, Opera, and Chrome and target FTP applications. In addition to that, we can observe a similar anti-analysis behavior. For example, debugger detection and hiding the process.

On the other hand, the XLoader macOS payload avoids being debugged by calling ptrace with PT_DENY_ATTACH and seems to target browsers only. Another difference is that XLoader macOS starts a copy of itself directly without injecting into system-related processes (Figure 3 bottom).

Both payloads try to be persistent on the running system either by using the registry key (Windows) or by creating a launch agent (macOS).



Create Directory	/Users/lxfjnlar	permission = 777₈
Create Directory	/Users	permission = 777₈
Create Directory	/Users/lxfjnlar/.rH6PLhr8yR/Bdch.app/Contents/MacOS	permission = 777₈
Create Directory	/Users/lxfjnlar/.rH6PLhr8yR/Bdch.app	permission = 777₈
Create Directory	/Users/lxfjnlar/.rH6PLhr8yR/Bdch.app/Contents	permission = 777₈
Create Directory	/Users/lxfjnlar/.rH6PLhr8yR	permission = 777₈
Get Info	/Users/lxfjnlar/.rH6PLhr8yR/Bdch.app/Contents	
Get Info	/Users/lxfjnlar/.rH6PLhr8yR/Bdch.app/Contents/MacOS	
Get Info	/Users/lxfjnlar/Hgl2uHXq	
Open	/Users/lxfjnlar/Hgl2uHXq	desired_access = O_NONBLOCK
Read	/Users/lxfjnlar/Hgl2uHXq	size = 127808, size_out = 127808
Write	/Users/lxfjnlar/Library/LaunchAgents/com.rH6PLhr8yR.Bdch.plist	size = 470
Write	/Users/lxfjnlar/.rH6PLhr8yR/Bdch.app/Contents/Info.plist	size = 777
Write	/Users/lxfjnlar/.rH6PLhr8yR/Bdch.app/Contents/MacOS/Bdch	size = 127808

Figure 5: VMRay Platform – Persistence mechanism on macOS.

By taking a look at the binary level, we see that XLoader Windows resolves API functions at runtime which is similar to FormBook. Another commonality is the usage of so-called encbufs. These encbufs are part of the code section and look at first sight like ordinary functions because they have a typical function prologue. However, they are used to store encrypted data. Both of these techniques complicate a manual analysis without a sandbox.

Moving forward to the network communication, both XLoader and FormBook use a specific byte pattern, also known as magic bytes, in the C2 communication which is build upon HTTP. These bytes are part of the messages transmitted during the communication. While FormBook uses “FBNG”, XLoader uses the magic bytes “XLNG”.

In addition to the network connection to their C2 servers, both connect to multiple benign URLs at runtime, which include the mission-id of the respective sample. We refer to them in the following as decoy URLs. Figure 6. shows the HTTP requests sent to one of the decoy URLs the sample connects to. As highlighted in the figure, it includes the mission-id (“xzes”)

despite being a decoy while the real C2 elsist[.]online is unreachable. This makes it challenging to identify the real C2 server from the decoy URLs based on network connection attempts because the attempts look similar.

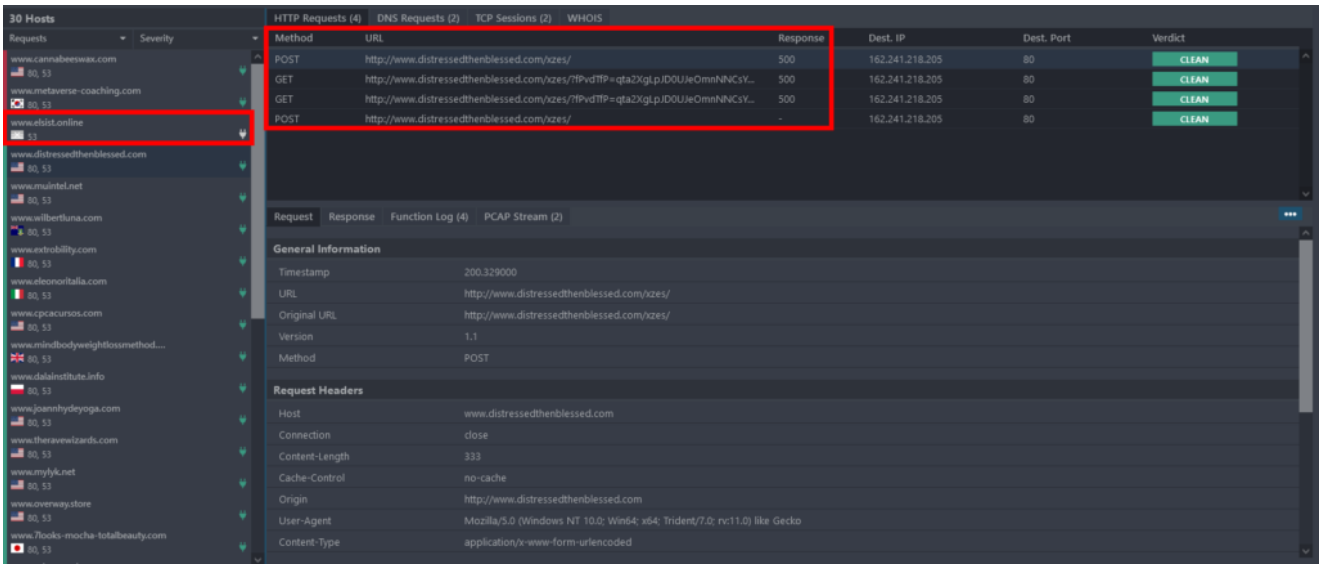


Figure 6: VMRay Platform – C2 Communication including decoys from the XLoader Windows payload.

This behavior makes it hard to separate the real C2 URL from the decoy URLs and complicates the automatic extraction of IOCs in general.

In such a case, the VMRay platform's artifact scoring feature assists analysts to distinguish artifacts from IOCs. It can mark the real C2 URL as an IOC while the decoy URLs remain artifacts. Therefore, the platform can extract the real network IOCs more reliable.

Conclusion

In this Spotlight, we discussed XLoader with a focus on the distribution with XBinder. The generated sample involves three different file types. Performing a dynamic analysis of the sample using a sandbox requires the sandbox supports these file types and respective operating systems.

Furthermore, FormBook and XLoader use decoy domains to make the C2 connection unapparent. Through the VMRay platform's custom IOC generation, the network IOC extraction of the real C2 URL is more reliable.

Resources

https://blogs.blackberry.com/en/2021/09/threat-thursday-xloader-infostealer

https://research.checkpoint.com/2021/time-proven-tricks-in-a-new-environment-the-macos-evolution-of-formbook/

https://www.netscout.com/blog/asert/formidable-formbook-form-grabber

IOCs

C2 URLs – www[.]elsist[.]online/xzes

-www[.]iregentos[.]info/09rb

SHA-256 XLoader macOS –
97d6b194da410db82d9974aec984cff8ac0a6ad59ec72b79d4b2a4672b5aa8aa

SHA-256 XLoader Windows –
4216ff4fa7533209a6e50c6f05c5216b8afb456e6a3ab6b65ed9fcbdbd275096

SHA-256 XBinder Builder –
693d6f0ac1e3f5e3e5b68c45d2a77bcc9d8976f7b091d5bfa1e719ad8b97fd25