More ProxyShell? Web Shells Lead to ZeroLogon and Application Impersonation Attacks

fortinet.com/blog/threat-research/more-proxyshell-web-shells-lead-to-zerologon-and-application-impersonation-attacks

September 14, 2021

FortiGuard Labs Threat Research Report

Affected platforms: Microsoft Exchange
Impacted parties: Exchange Mailboxes
Impact: Gives unauthorized users the ability to access and send emails from any user within the organization
Severity level: Critical

Special thanks to Angelo Cris Deveraturda, Wilson Agad, Llallum Victoria, Wil Vidal, Jared Betts, and Ken Evans.

Introduction of ProxyShell

FortiGuard Labs recently discovered an unidentified threat actor leveraging ProxyShell exploits using techniques that have yet to be reported. Multiple instances of FortiEDR had detected malicious DLLs in memory, and we uncovered these new techniques while consulting with one of the organizations that had been compromised by ProxyShell. Through active threat hunting, we were then able to determine that other organizations had also been compromised.

The DLLs, which were previously unknown based on their SHA256 file hashes, were used to perform active reconnaissance, obtain hashed passwords via Zerologon, and perform pass-the-hash authentication to establish persistence via Exchange Application Impersonation. This blog intends to provide an analysis of these DLLs. We documented the malicious activity

associated with them by recreating the incidents in a lab environment. The goal is to help the public and future customers determine if they have related activity in their environment and take appropriate action.

Overview of ProxyShell Incidents

These events began around the time that ProxyShell hit the cyber news headlines. At first, they seemed to match what most organizations were already reporting. Exploit details, from the directories to the types of web shells used, matched almost verbatim. The difference was when web shells performed post-exploitation activity via DLLs loaded into memory, which triggered events within FortiEDR.

FortiEDR detected these DLLs because they loaded into memory space allocated for vbc.exe, the Visual Basic Compiler for .NET applications, and were loaded from the w3wp.exe process, which is used to run Microsoft Exchange's Outlook Web Application. This, along with FortiEDR's machine learning algorithm, determined that these files were likely malicious.

The figure below shows w3wp.exe injecting a thread into the vbc.exe process and accessing services on the Exchange server.

Figure 1: w3wp.exe Injecting a remote thread and accessing services on an Exchange server

Next, the web shells were used to connect to numerous domain controllers within the environment, and we found the Thor.dll file in memory. This DLL is used to authenticate to the domain, modify security permissions, create domain accounts, and add accounts to Exchange roles. I analyze the Thor.dll in detail later.

Figure 2: w3wp.exe loading the Thor.dll into memory

The event shown in Figure 2 occurred just after an HTTP POST was sent to one of the web shells and seconds before a domain account was created within the environment. A legitimate domain service account created the account within the organization. Of note, when FortiEDR was in protection mode, all this activity was blocked. No domain accounts or further activity was seen in those situations.

Also, the domain accounts for the other environments were never used for interactive authentication, which told us that the threat actor was likely interested in using the accounts later in this campaign. They also only had domain user access with no elevated privileges, indicating they were not interested in using the accounts to perform administrative level tasks.

To fully understand what took place when these DLLs were loaded into memory, we analyzed their functionality and behavior, explained in detail in the following section.

DLL Analysis

There was a total of 22 DLLs found in memory by FortiEDR during the time of these events. Not all were tagged as malicious, but FortiEDR captured them regardless as they were related to the malicious events. Fortunately, all these DLLs were .NET compiled, so decompiling them was relatively trivial.

The first thing that was captured was the original names of the DLLs.

Figure 3: DLL Original DLL names when decompiled

Some of the names appear to be randomly named; however, others are named overtly with their intended function. Further analysis shows that the randomly named DLLs were loaders for the overtly named DLLs. This can be seen by looking within the "xml()" method of the decompiled code for each loader DLL.

Figure 4: First portion of the xml() method to show loader functionality

The "xml()" method loads the base64 version of the DLL and calls a method within the DLL. In this example, it was the "Elite" method within the "Axiomatic" class.

The loader DLLs are also responsible for taking the output from the method calls and relaying it back to the web shell HTTP session that it was called from so the threat actor will get the output results from the DLL execution. This provides the threat actor with results from the HTTP session from which they interacted with the web shell.

Figure 5: The method call for HTTP response

By using these method calls and linking them to the loaded DLLs, our team could recreate this activity using PowerShell. The results are shown below and are broken down by each DLL, along with a screenshot showing the output sent back through the web shell HTTP session.

GetEcpWebConfigModule.dll

This DLL gets the Exchange Control Panel (ECP) Web.config file from the Exchange server. This gives the threat actor access to some of the internal ECP configurations. More information about the web.config file can be found here.

Figure 6: Output of executing GetEcpWebConfigModule.dll

DomainGroups.dll

This DLL gets all the domain groups from Active Directory. This gives the threat actor information about the hierarchy of the environment in which they are working.

Figure 7: Output of executing DomainGroups.dll

DnsDump.dll

This DLL gets the DNS Type A entries, which can be used to target additional machines within the environment. It also provides the threat actor with a forward and reverse lookup database when translating hostname to IP and vice-versa.

Figure 8: Output of executing DnsDump.dll

OrganizationManagement.dll

This DLL gets the list of users in the user group "Organizational Management." Members within this group have the necessary access on the Exchange server to create and/or modify roles and permissions.

Figure 9: Output of executing OrganizationManagement.dll

ExchangeServers.dll

This DLL gets the list of exchange servers within the domain so the threat actor can target other Exchange servers if needed. Some environments are set up in a hierarchy, where different roles of Exchange are on different servers. A threat actor would need this information to know which servers to target.

Figure 10: Output of executing ExchangeServers.dll

DiskInfo.dll

This DLL gets information about the drives of the localhost. This information can be useful to the threat actor to determine which hard drive may contain Exchange databases.

Figure 11: Output of executing DiskInfo.dll

Computers.dll

This DLL gets a list of workstations within the domain, which gives the threat actor an idea of how large an environment is and a good starting point for launching additional attacks.

Figure 12: Output of executing Computers.dll

SystemInfoModule.dll

This DLL gets information about the system, such as build name, hardware info, OS, etc. This provides the threat actor with the necessary information to launch further exploitation attacks or maintain persistence and access.

Figure 13: Output of executing SystemInfoModule.dll

SystemService.dll

This DLL returns a list of services running on the local system, which gives the threat actor a list of security software present on the system. For example, the threat actor may have to use bypass techniques to avoid certain types of security products.

Figure 14: SystemService.dll

Machine.Modules.ZerologonInterface.dll

This DLL reaches out to given domain controllers (passed as a parameter) to see if they're vulnerable to Zerologon, and if so, responds with usernames and hashed passwords. It then restores the computer hash for the domain controller to its original state. This is a documented method using Zerologon and can be read about more here under the section "Case 2 – DC Password Reset with Original Password Reestablished".

Figure 15: Output of executing Machine.Modules.ZerologonInterface.dll

Thor.dll

This DLL attempts to authenticate with credentials from the previous Zerologon dump and create domain accounts attempting to masquerade as healthmailbox accounts. It also creates a role-based access control (RBAC) assignment within Exchange that gives it the ApplicationImpersonation role built into Microsoft Exchange. This allows the actor to access and send email from any mailbox within the environment. Furthermore, it locks down the permissions of those objects.

Figure 16: Output of executing Thor.dll and descriptions of output

We can also see the role assignment created within Exchange, along with the role type, "ApplicationImpersonation".

Figure 17: Exchange role policy output with threat actor-created role

Figure 18: ApplicationImpersonation role type added to a role

ApplicationImpersonation is meant to be used by service applications that need to access multiple mailboxes and "act as" the mailbox owner. However, in this case, it allows the attacker to retain access to the domain's mailboxes. More information about Application Impersonation using EWS can be found on Microsoft's documentation page (https://docs.microsoft.com/en-us/exchange/client-developer/exchange-web-services/impersonation-and-ews-in-exchange). This specific email compromise technique is rarely seen, and the intent behind this activity is still largely unknown as the threat actor wasn't seen using the created accounts to impersonate mailboxes in a victim's environment.

ProxyShell Conclusion

FortiEDR was able to capture DLLs found in memory that had been executed via web shells created from ProxyShell vulnerabilities. In the situations where the DLLs were not blocked (simulation blocks), there was no follow-on activity, nor was an intent identified. But the threat actors were clearly interested in maintaining access to the organization's mailboxes using techniques rarely used and largely undocumented.

Fortinet Protections

FortiEDR detects and blocks (when enabled) the malicious DLLs seen in memory.

All payloads described are detected and blocked by the FortiGuard AntiVirus service.

IOCs

SHA256 Hashes and DLLs

Computers.dll	EB4EF1B588AD5027409B4D6B5643E25E456B14A5175EFFCF866B31C2CCD35FC3
DiskInfo.dll	2AF411EA82384F7C41954E5CE4755709E80DA75C57357D62208018137111DDD3
DnsDump.dll	F1E4E139EA33385E672A7EB35BF5B33570709FB004B5C6884D12610D28C94797
DomainGroups.dll	3112BE27CB28AEDD9334567A0E24569EC3D5F75414539E17433E4B4F27AA9470
ExchangeServers.dll	5E0FE024A39D0AF4F045DC6770A00F4E9FFBAB1D71C68360331CF8A8BB16BC18

GetEcpWebConfigModule.dll	CDD79F442519B0F48DDC7A1A699DFCE8E597445975F84BFAC51A2F0BA80F1237
Machine.Modules.ZerologonInterface.dll	A46B18A07B1509B15EB7FC6CF876888DC780962A3F850DC3B3EE5C7FBBEE723C
OrganizationManagement.dll	28D983C235F2C5420AE596CDC3AAEF65D953E76F935B73708DA8E47072ED66B8
SystemInfoModule.dll	FC73324432E7BB3FAAF08527E462D26D0E4B3D1095BF7CA09BA04DD5B4E71664
SystemService.dll	4E8EA9B3261B49597EF6D37DD252B37CA2BD162713B900EA25A65CCDB9B98DE5
Thor.dll	AFD7E5431E86BAA55A6FDCB8CE81F1E7EEA158BA388D08C59678789AF6FE7C0E

Table 1: Names and SHA256 Hashes for DLLs

Account Names

Healthmailbox<random 7 alphanumeric characters>

Learn more about Fortinet's FortiGuard Labs threat research and intelligence organization and the FortiGuard Security Subscriptions and Services portfolio.

Learn more about Fortinet's free cybersecurity training, an initiative of Fortinet's Training Advancement Agenda (TAA), or about the Fortinet Network Security Expert program, Security Academy program, and Veterans program. Learn more about FortiGuard Labs global threat intelligence and research and the FortiGuard Security Subscriptions and Services portfolio.