BOLETIM DE SEGURANÇA

Grupo APT-C-36 tem como alvo indústrias de manufatura

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

Recentemente a eSentire informou sobre o Blind Eagle, identificado como APT-C-36, surgindo em 2018 e originário da América do Sul, tem como alvo a Colômbia e outros países vizinhos. Utiliza e-mails de phishing para iniciar seus ataques. Em 2021, a Trend Micro destacou em um blog que o Blind Eagle implantou várias variantes de RAT, incluindo **njRAT**, **Remcos**, **Imminent Monitor**, **AsyncRAT**, **LimeRAT**, **BitRAT** e **Warzone RAT**. Notou-se que o Blind Eagle tem como alvo, indústrias de manufatura. Enviando e-mail de phishing com um link para baixar arquivos RAR e BZ2 contendo um arquivo VBS malicioso para os usuários.

2 ANÁLISE DO ANDE LOADER

O arquivo RAR, que é protegido por senha, abrigando um arquivo VBS malicioso. Este arquivo VBS tem um código que copia o próprio arquivo para a pasta *Startup* (**\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup**) para garantir a persistência, utilizando o método **file.Copy**. Antes de realizar a cópia, um atraso é introduzido com o comando "**cmd.exe /c ping 127.0.0.1 -n 10**". Mais adiante no script, existe um código ofuscado que usa um simples "Replace" para conter um comando do PowerShell codificado em base64. Este comando carrega um assembly (**$rOWg**), recupera um tipo específico (**Fiber.Home**) e invoca um método (**VAI**) nesse tipo. A invocação do método é feita com uma matriz de parâmetros.

Figura 1 – Comando ofuscado do Powershell.

Figura 2 – Comando contendo matriz de parâmentros.

Com mais detalhes, na imagem abaixo mostra a cadeia de infecção.

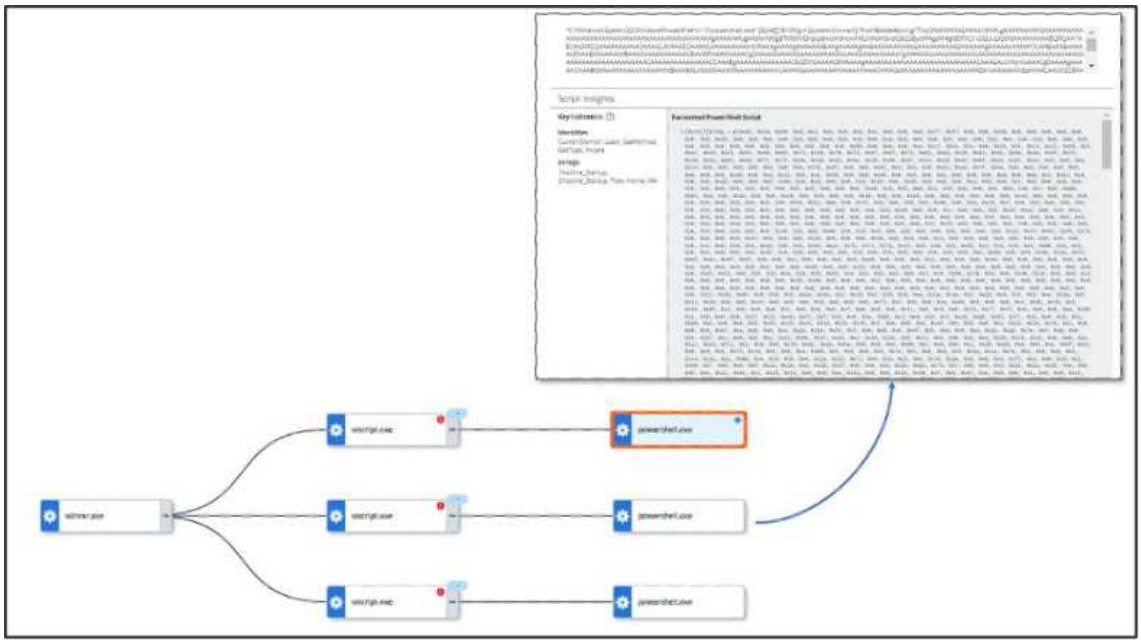

Figura 3 – Cadeia de infecção

Ao decifrar o comando em Base64, foi encontrado um binário .NET (**MD5: 48b6064beec687fc110145cf7a19640d**) que foi ofuscado com o *YanoObfuscator* versão 1.0.15.0. A função de descriptografia aplica operações XOR e bit a bit em cada caractere da string de entrada, usando uma chave variável (num) baseada em um número inteiro fornecido. Os caracteres alterados são armazenados em um array, que é então convertido de volta em uma string e retornada como o resultado descriptografado.

A descriptografia funciona quando, a função inicializa duas variáveis: num é definido como **356636782 + A_1** e num2 é definido como 0. A string de entrada A_0 é convertida em um array de caracteres chamado array. A função entra em um loop. Em cada iteração do loop, ele executa as seguintes ações: Verifica o valor de num2 para determinar a ação apropriada. Se num2 for 0, define num3 como 0 e prossegue para a próxima etapa. Se num2 for 1, inicializa num3 como 0. Se num2 for 2, incrementa num3 e prossegue para a próxima etapa. Se num2 for 3, pula para a próxima iteração do loop. Se num3 for maior ou igual ao comprimento do array, ele sai do loop. Caso contrário, executa algumas operações bit a bit no caractere no índice num3 do array: Aplica uma operação XOR entre os 8 bits inferiores do caractere e o valor atual de num. Ele desloca o valor resultante 8 bits para a esquerda e o combina com a operação XOR entre os 8 bits superiores do caractere e o valor incrementado de num. O valor resultante é armazenado de volta no array no mesmo índice. Define num2 como 2 para continuar o loop. Após o término do loop, o array modificado é convertido novamente em uma string usando o construtor de string. A string resultante é então retornada como o valor descriptografado.

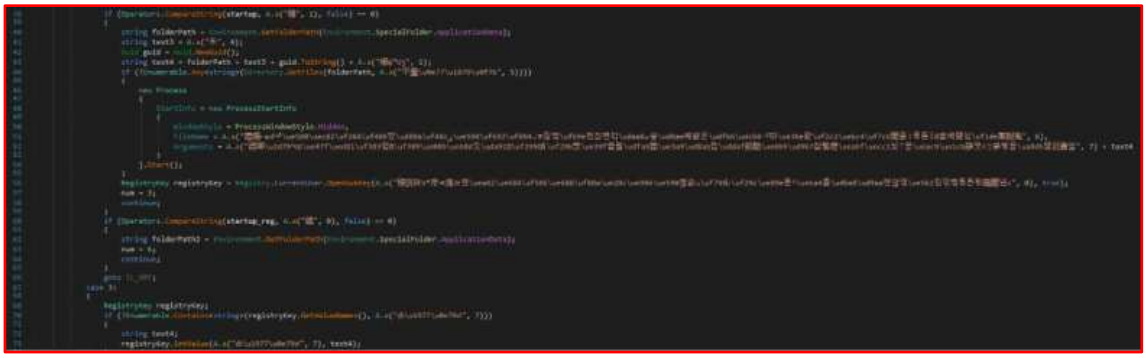

Figura 4 – Strings ofuscadas

```
// Token: 0x02000019 RID: 25
internal class A
{
    // Token: 0x0600004A RID: 74 RVA: 0x000030C4 File Offset: 0x000012C4
    [MethodImpl(MethodImplOptions.NoInlining)]
    internal static string a(string A_0, int A_1)
    {
        int num = 356636782 + A_1;
        int num2 = 0;
        char[] array;
        for (;;)
        {
            int num3;
            switch (num2)
            {
            case 0:
                array = A_0.ToCharArray();
                num2 = 1;
                continue;
            case 1:
                num3 = 0;
                break;
            case 2:
                num3++;
                num2 = 3;
                continue;
            }
            if (num3 >= array.Length)
            {
                break;
            }
            char[] array2 = array;
            int num4 = num3;
            char c = array[num3];
            array2[num4] = (ushort)((((int)(c & '9') ^ num++) << 8) | (int)((byte)((int)(c >> 8) ^ num++)));
            num2 = 2;
        }
        return string.Intern(new string(array));
    }
}
```

Figura 5 – Algoritmo de descriptografia

3 CRIPTOGRAFIA BYRODA

Foi divulgado o criptografador empregado em uma das campanhas do Blind Eagle. O criador do criptografador é conhecido nos fóruns de hackers como “Roda-Modder” ou simplesmente “Roda”. Além disso, desde 2014, esse desenvolvedor tem compartilhado outros criptografadores e protetores nesses fóruns.

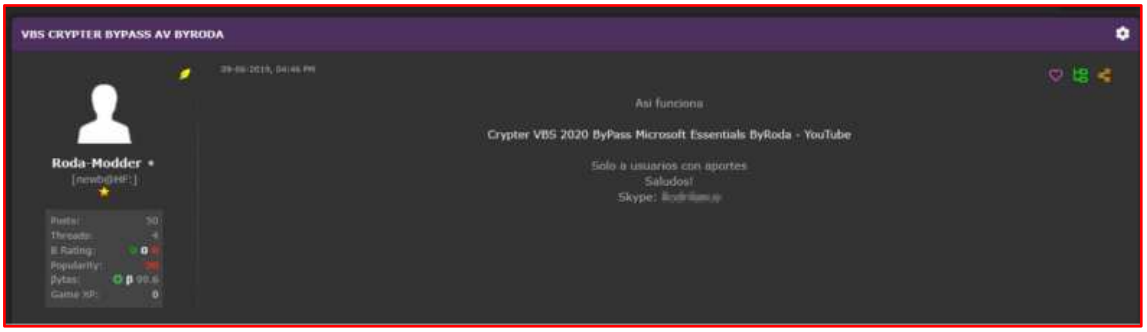

Figura 6 – Anúncio 1 do Cryptoter

Figura 7 – Anúncio 2 Cryptopter

Para que o criptografador seja ativado, o usuário precisaria fornecer a chave “ativa”.

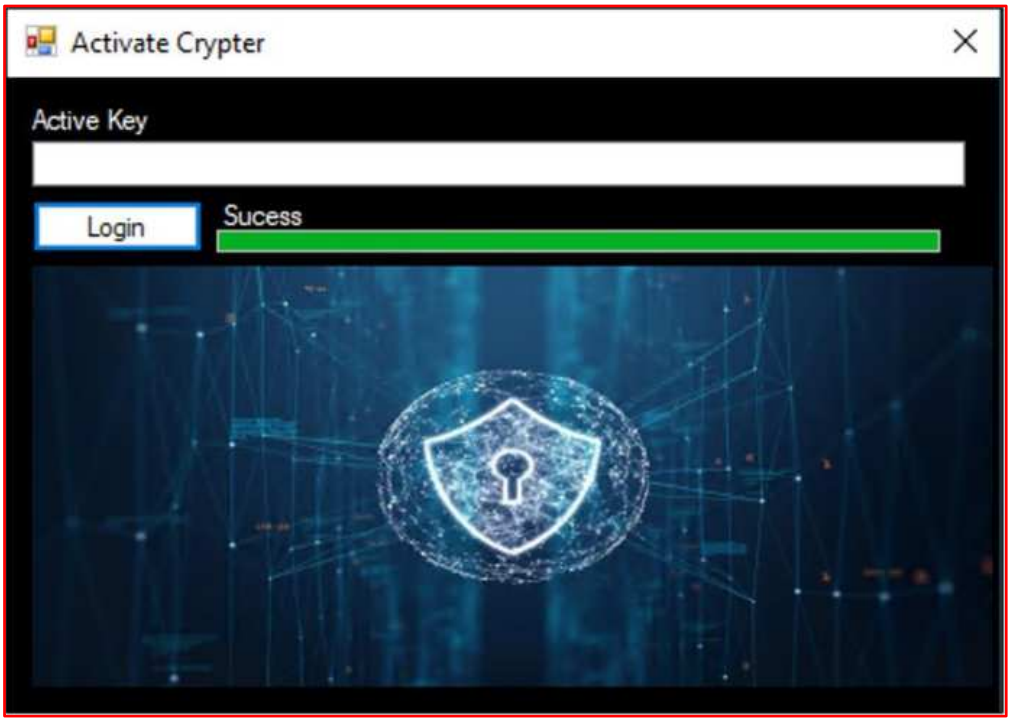

Figura 8 – Formulário de ativação do Crypter

O processo de ativação da chave ocorre quando a string, que está codificada em base64, é obtida do repositório GitHub do desenvolvedor. Quando o formulário é carregado, o método **Login_Load** é acionado e inicia o download da string do repositório GitHub. Após a conclusão do download iniciado pelo Login_Load, o método **x1_DownloadStringCompleted** é ativado. Este método invoca a função de descriptografia na string baixada para recuperar as chaves originais.

```
311         // Token: 0x06000087 RID: 135 RVA: 0x00004920 File Offset: 0x00002B20
312         private void x1_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs e)
313         {
314             try
315             {
316                 this.S = this.descrypted(e.Result);
317                 this.Label2.Text = "Sucess";
318                 this.TextBox1.Enabled = true;
319                 this.Button1.Enabled = true;
320             }
321             catch (Exception ex)
322             {
323                 Interaction.MsgBox(ex.ToString(), MsgBoxStyle.OkOnly, null);
324             }
```

Figura 9 – método x1_DownloadStringCompleted

O método “**descrypted**” é acionado em “**x1_DownloadStringCompleted**” para decodificar a string baixada de Base64. Ele pega a string codificada em base64, substitui “@” por “1”, realiza a decodificação em base64 e depois inverte a string. A string resultante, agora descriptografada, é armazenada em “this.S” para uso posterior no método “Button1_Click”.

```
327         // Token: 0x06000088 RID: 136 RVA: 0x0000499C File Offset: 0x0000289C
328         private string descrypted(string Key)
329         {
330             string text = Key.Replace("@", "1");
331             return Strings.StrReverse(Encoding.ASCII.GetString(Convert.FromBase64String(text)));
```

Figura 10 – Método descrypted

O método "**Button1_Click**" é acionado quando o usuário clica no botão. Ele divide a string descriptografada em um array de chaves usando "//" como delimitador e compara cada chave com o texto inserido pelo usuário. Se uma correspondência é encontrada, o usuário é autenticado e uma mensagem de sucesso é exibida. Se a correspondência não é encontrada, o usuário recebe uma mensagem pop-up "Chave expirada!!".

```
281         // Token: 0x06000086 RID: 134 RVA: 0x00004850 File Offset: 0x00002A50
282         private void Button1_Click(object sender, EventArgs e)
283         {
284             checked
285             {
286                 try
287                 {
288                     string[] array = Strings.Split(this.S, "//", -1, CompareMethod.Binary);
289                     int num = Enumerable.Count<string>(array) - 2;
290                     for (int i = 0; i <= num; i++)
291                     {
292                         string text = array[i].Trim();
293                         bool flag = Operators.CompareString(text.ToLower(), this.TextBox1.Text.ToLower().Trim(), false) == 0;
294                         if (flag)
295                         {
296                             Interaction.MsgBox("Sucess", MsgBoxStyle.OkOnly, null);
297                             MyProject.Forms.Form1.Show();
298                             base.Hide();
299                             return;
300                         }
301                     }
302                     Interaction.MsgBox("Expired key!!", MsgBoxStyle.OkOnly, null);
303                 }
304                 catch (Exception ex)
305                 {
306                     Interaction.MsgBox(ex.ToString(), MsgBoxStyle.OkOnly, null);
307                 }
308             }
```

Figura 11 – Método Button1_Click.

Figura 12 – Painel do FuckCrypt

O criptografador pode ser produzido em extensões VBS e JS, com opções de persistência como nome de inicialização, tarefa programada e AntiVM. A carga útil é enviada para o Pastebin e, em seguida, para o **pasteio[.]com** para recuperar o injetor. Observamos uma versão diferente do criptografador postada por um pesquisador de segurança, **@1ZRR4H**. O criptografador é enviado para o **Pastebin** e, em seguida, para o **wtools[.]io** para recuperar os componentes do injetor. No momento da escrita deste blog, o pasteio parece estar indisponível, tornando a versão 2.1 do FuckCrypt inoperante. O VBS gerado contém uma única linha do PowerShell codificada em base64 ofuscada e um código indesejado que pode ser encontrado codificado na seção de Recursos do criptografador.

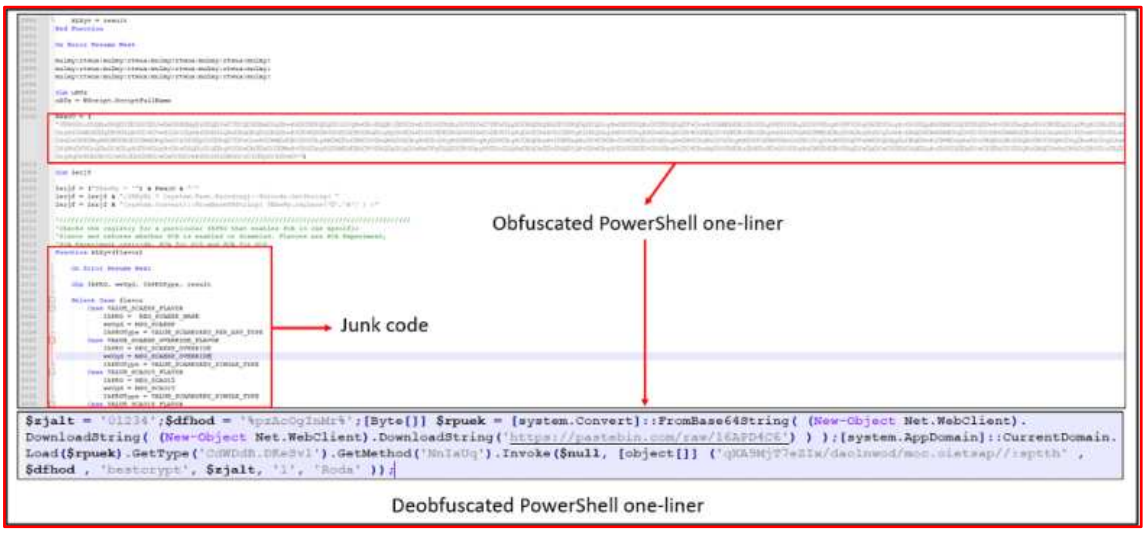

Figura 13 – Arquivo VBS criptografado

Figura 14 – Seção Recursos Incorporados

O primeiro arquivo que foi baixado, ele é parcialmente responsável pela injeção do processo e são utilizadas funções como **GetThreadContext**, **SetThreadContext**, **ReadProcessMemory**, **NtUnmapViewOfSection**, **VirtualAllocEx**, **ResumeThread**, entre outras, indicando um esvaziamento de processo. A DLL também inclui outras APIs que são conhecidas por serem utilizadas na injeção de processos. Com base no códig, a carga final decodificada em base64 seria injetada no InstalUtil.exe.

Figura 15 – Seção Recursos

Se o comando único do PowerShell decodificado tiver “4” como quarto parâmetro passado para o binário, isso indica que o AntiVM está ativado. O recurso AntiVM procura por processos como vmtoolsd ou VirtualBox rodando na máquina comprometida. Se um processo com qualquer um desses nomes for detectado (o que sugere que uma máquina virtual está atualmente em operação), a função será interrompida e retornará imediatamente.

Figura 16 – Verificação da VM

Caso o quarto parâmetro também tenha o valor “1”, o código irá gerar uma nova entrada no registro em “**HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**”. Essa entrada contém um valor que executa um comando do PowerShell para rodar um arquivo VBScript localizado na pasta %TEMP%. Dessa maneira, o código é executado sempre que o sistema é iniciado. O arquivo que contém o comando do PowerShell é chamado de “**xx1.ps1**”. Posteriormente, o código cria um comando VBScript que executa o script VBS criptografado inicial em um novo arquivo chamado “**xx2.vbs**” na pasta %TEMP%.

Figura 17 – Condição 1 do quarto parâmetro

Um criptografador adicional (**MD5: b167a0bc7b097550a89a5ba4cb258592**), criado por Roda, retira os componentes extras do injetor do servidor codificado. Com um nível médio de confiança, foi avaliado que o desenvolvedor FuckCrypt também participa da campanha Blind Eagle, descartando o malware que está armazenado no mesmo servidor.

Figura 18 – FuckCrypt

Figura 19 – IP codificado contendo componentes do injetor

Também foi possível encontrar outras amostras associadas ao binário ou ao desenvolvedor.

4 MITRE ATT&CK - TTPs

Tática	Técnica	Detalhes
Initial Access	T1566	Os adversários podem enviar mensagens de phishing para obter acesso aos sistemas das vítimas. Todas as formas de phishing são entregues eletronicamente por engenharia social.
User Execution	T1204.002	Um adversário pode contar com a abertura de um arquivo malicioso pelo usuário para obter execução.
Persistence	T1547.001	Os adversários podem obter persistência adicionando um programa a uma pasta de inicialização ou referenciando-o com uma chave de execução do Registro.
Execution	T1059.001	Os adversários podem abusar de comandos e scripts do PowerShell para execução.
Defense Evasion, Privilege Escalation	T1055.012	Os adversários podem injetar código malicioso em processos suspensos e vazios para escapar das defesas baseadas no processo.

Tabela 1 – Tabela MITRE ATT&CK.

5 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:

Conscientização

- Eduque-se e a sua equipe sobre as táticas, técnicas e procedimentos usados por atores de ameaças como o Blind Eagle. Isso inclui o reconhecimento de e-mails de phishing e a importância de não clicar em links ou abrir anexos suspeitos.

Atualizações de segurança

- Mantenha todos os sistemas operacionais e softwares atualizados. Muitos atores de ameaças exploram vulnerabilidades conhecidas em softwares desatualizados.

Software antivírus

- Use um software antivírus confiável e mantenha-o atualizado. Além disso, configure-o para realizar varreduras regulares.

Backups

- Faça backups regulares de seus dados importantes e garanta que esses backups estejam seguros e possam ser restaurados.

Restrição de privilégios

- Implemente o princípio do menor privilégio. Os usuários devem ter apenas os privilégios necessários para realizar suas tarefas e nada mais.

Monitoramento de rede

- Monitore regularmente a rede em busca de atividades suspeitas. Isso pode ajudar a detectar uma infecção em seus estágios iniciais.

Planos de resposta a incidentes

- Tenha um plano de resposta a incidentes em vigor. Isso deve incluir a identificação de quem em sua organização deve ser notificado em caso de um incidente.

6 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:	0b9cc70477af81a3fc8a5d335162f96d
sha1:	d527c4a9e64c20af2e3854e9c3d2792de5d19e83
sha256:	4f44ee43c3b1aa0a3654d9a93972acd198fae39c3ab71f3f1f2f1302771fa365
File name:	Vbs-Crypter Simples.exe

Indicadores de compromisso do artefato	
md5:	b167a0bc7b097550a89a5ba4cb258592
sha1:	1c33e07d7c044863fedc20c1ab14d7aadec7d14c
sha256:	87effdf835590f85db589768b14adae2f76b59b2f33fae0300aef50575e6340d
File name:	FuckCrypt.exe

Indicadores de compromisso do artefato	
md5:	191d5bf5d3ab54549d436399bcab642d
sha1:	d0000ad31f31f89684a4bdbbdc2bfec67f342400
sha256:	c5b11f830602e641f7d86a756da6b745d80ef6431be3f373be6912cab5f7acf5
File name:	Vbs-Crypter.exe

Indicadores de compromisso do artefato	
md5:	137f21d1f8fdd5cfe86637368b526027
sha1:	a8bf076482b60609b77ee379bade5490b47267c8
sha256:	8b6a909110ca907eb279cfb8f6db432af5564263e49c6982001b83fcffe04c07
File name:	rm-reversed.exe.vir

Indicadores de compromisso do artefato	
md5:	7b72f2775b7bf33c9778533480d34e04
sha1:	42b9a9ac5ecf0d6c03de38d204926a79aeff8de8
sha256:	54716a9a3a8fb7cc6be3074ea0472703ec03e1421d553b0dc6b3ebe7b1ec10bb
File name:	njs.exe

Tabela 2 – Indicadores de Compromissos de artefatos

Indicadores de URL, IPs e Domínios

Indicadores de URL, IPs e Domínios	
URL	rxms.duckdns[.]org:57832
Domínio	njnjnjs[.]duckdns.org
IP	91.213.50[.]74

Tabela 3 – 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.

7 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- **eSentire**
- **Thehackernews**

heimdall
security research
A DIVISION OF ISH