BOLETIM DE SEGURANÇA

APT41 aprimora arsenal de malware com adições do DodgeBox e MoonWalk

heimdall
security research
A DIVISION OF ISH

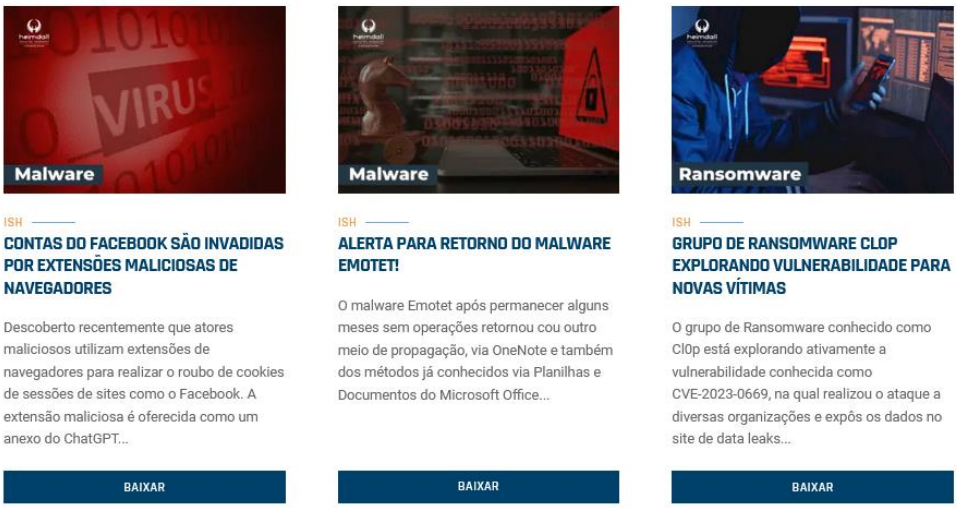
TLP: CLEAR

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

O grupo APT41, associado à China e conhecido por suas ameaças persistentes avançadas (APT), está supostamente empregando uma versão aprimorada do malware StealthVector. Esta versão atualizada é usada para implantar um backdoor inédito, denominado MoonWalk.

2 INFORMAÇÕES SOBRE A AMEAÇA

O grupo APT41 utiliza o sideloading de DLL para operar o DodgeBox. Eles fazem uso de um executável legítimo (taskhost.exe), assinado pelo Sandboxie, para realizar o sideload de uma DLL maliciosa (sbiedll.dll). Esta DLL maliciosa, conhecida como DodgeBox, atua como um carregador e é encarregada de descriptografar uma carga útil de segundo estágio de um arquivo DAT criptografado (sbiedll.dat). A carga útil descriptografada, MoonWalk, atua como um backdoor que explora o Google Drive para comunicação de comando e controle (C2). A figura abaixo mostra a cadeia de ataque em um nível alto.

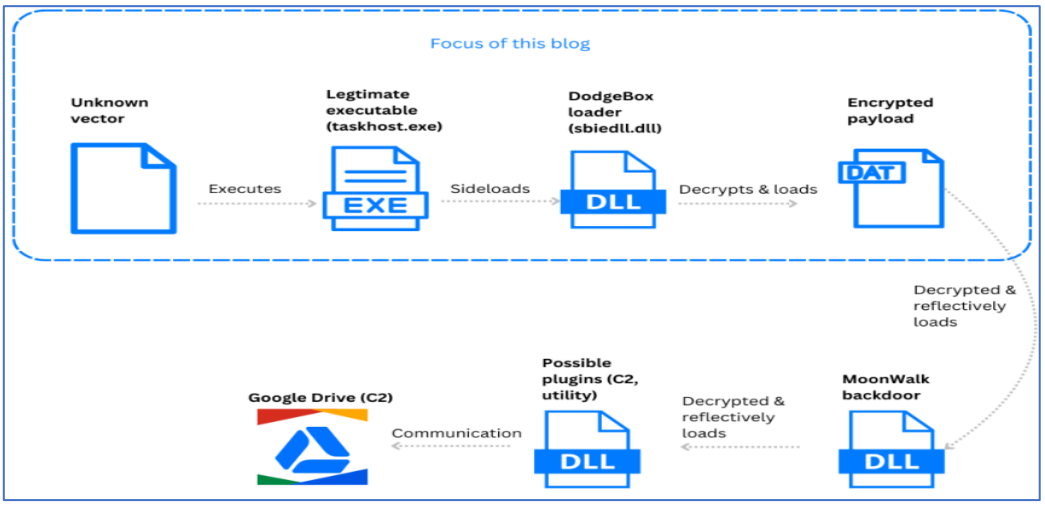


Figura 1 – Cadeia de ataque do DodgeBox e do MoonWalk.

O DodgeBox, um carregador de DLL reflexivo escrito em C, apresenta semelhanças com o StealthVector em termos de conceito, mas incorpora melhorias significativas em sua implementação. Ele oferece vários recursos, incluindo descriptografia e carregamento de DLLs incorporadas, realização de verificações e ligações de ambiente e execução de procedimentos de limpeza. O que diferencia o DodgeBox de outros malwares são seus algoritmos e técnicas exclusivos.

Durante nossas atividades de caça a ameaças, encontramos duas amostras do DodgeBox que foram projetadas para serem carregadas lateralmente por executáveis legítimos assinados. Um desses executáveis foi desenvolvido pela Sandboxie (SandboxieWUAU.exe), enquanto o outro foi desenvolvido pela AhnLab. Todas as exportações dentro da DLL apontam para uma única função que invoca principalmente a função principal do malware, conforme ilustrado abaixo:

```
void SbieDll_Hook()
{
 if ( dwExportCalled )
 {
   Sleep(0xFFFFFFFF);
 }
 else
 {
   hSbieDll_ = hSbieDll;
   dwExportCalled = 1;
   MalwareMain();
 }
}
```

Figura 2 – Exportação de DLL.

O MalwareMain implementa a funcionalidade principal do DodgeBox e pode ser dividido em três fases principais:

Descriptografia da configuração do DodgeBox

- O DodgeBox emprega o modo AES Cipher Feedback (AES-CFB) para criptografar sua configuração. O AES-CFB transforma o AES de uma cifra de bloco em uma cifra de fluxo, permitindo a criptografia de dados com diferentes comprimentos sem exigir preenchimento. A configuração criptografada é incorporada na .dataseção do binário. Para garantir a integridade da configuração, o DodgeBox utiliza hashes MD5 codificados para validar as chaves AES incorporadas e a configuração criptografada. Para referência, um exemplo da configuração descriptografada do DodgeBox pode ser encontrado na seção Apêndice deste blog. Referenciaremos esta configuração de exemplo usando a variável Configna seções a seguir.

Guarda-corpos de execução e configuração do ambiente

- Após descriptografar sua configuração, o DodgeBox executa diversas verificações de ambiente para garantir que esteja sendo executado no alvo pretendido. O DodgeBox começa verificando se o processo foi iniciado com os argumentos corretos. Ele verifica o argvparâmetro para uma string específica definida em Config.szArgFlag. Em seguida, ele calcula o hash MD5 do argumento subsequente e o compara ao hash especificado em Config.rgbArgFlagValueMD5. Nesse caso, o DodgeBox espera que os argumentos incluam --type driver. Se essa verificação falhar, o processo será encerrado.

Descriptografia de carga útil e codificação de ambiente

- Na fase final, o DodgeBox inicia o processo de descriptografia para o arquivo DAT de payload do MoonWalk. O código começa inspecionando os quatro primeiros bytes do arquivo. Se esses bytes forem diferentes de zero, isso significa que o arquivo DAT foi vinculado a uma máquina específica (o que é descrito abaixo). No entanto, se o arquivo DAT não for específico da máquina, o DodgeBox prossegue para descriptografar o arquivo usando criptografia AES-CFB, utilizando os parâmetros de chave armazenados no arquivo de configuração. Nas amostras analisadas pelo ThreatLabz, esse arquivo DAT descriptografado corresponde a uma DLL, que é o backdoor do MoonWalk. Após o processo de descriptografia, o DodgeBox dá o passo adicional de chavear a carga útil para a máquina atual. Ele faz isso criptografando novamente a carga útil usando a chave Config.rgbAESKeyForDatFile. No entanto, neste cenário específico, o processo desvia do IV (Initialization Vector) do arquivo de configuração. Em vez disso, ele utiliza o hash MD5 do GUID da máquina atual como o AES IV.

Esta abordagem garante que o arquivo DAT descriptografado não possa ser descriptografado em nenhuma outra máquina, aumentando assim a segurança da carga útil. Em seguida, o DodgeBox carrega reflexivamente a carga útil usando uma técnica de esvaziamento de DLL. Em um alto nível, o processo começa com a seleção aleatória de uma DLL host do System32diretório, garantindo que ela não esteja em uma lista de bloqueio (lista de bloqueio de DLL disponível na seção Apêndice) e tenha uma seção .text suficientemente grande. Uma cópia dessa DLL é então criada em C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Data.Trace\v4.0_4.0.0.0__<random bytes from pcrt4!UuidCreate><name of chosen DLL>.dll. O DodgeBox modifica essa cópia desabilitando o sinalizador NX, removendo as seções reloce TLSe corrigindo seu ponto de entrada com um simples return 1.

Após a preparação da DLL do host para injeção, o DodgeBox prossegue zerando os cabeçalhos PE e as IMAGE_DATA_DIRECTORYestruturas correspondentes aos diretórios import, reloc, e debugda DLL de carga útil. Essa DLL de carga útil modificada é então inserida na DLL do host selecionada anteriormente. A cópia resultante da DLL do host modificada é carregada na memória usando as APIs NtCreateSectione NtMapViewOfSection.

Depois que a DLL é carregada com sucesso, o DodgeBox atualiza as entradas relevantes no Process Environment Block (PEB) para refletir a DLL recém-carregada. Para ocultar ainda mais suas atividades, o DodgeBox substitui a cópia modificada da DLL do host com seu conteúdo original, fazendo com que ela apareça como uma DLL legítima e assinada no disco. Finalmente, o malware chama o ponto de entrada da DLL de carga útil.

Curiosamente, se a função responsável pelo DLL hollowing falhar ao carregar a DLL de payload, o DodgeBox emprega um mecanismo de fallback. Essa função de fallback implementa uma forma tradicional de carregamento de DLL reflexivo usando NtAllocateVirtualMemorye NtProtectVirtualMemory. Nesse estágio, a DLL de carga útil foi carregada com sucesso e o controle é transferido para a DLL de carga útil invocando a primeira função exportada.

O DodgeBox, em todas as suas fases, emprega uma técnica final: o spoofing de pilha de chamadas. Esta técnica é usada para mascarar as origens das chamadas de API, tornando mais difícil para os EDRs e sistemas antivírus detectar atividades maliciosas. Ao alterar a pilha de chamadas, o DodgeBox faz com que as chamadas de API pareçam vir de binários confiáveis, em vez do próprio malware. Isso impede que as soluções de segurança obtenham informações contextuais sobre a verdadeira fonte das chamadas de API.

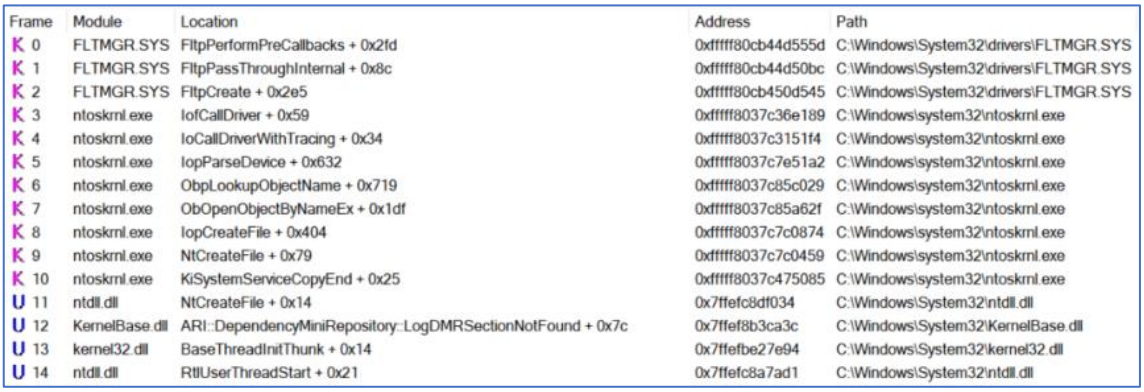
O malware usa especificamente o spoofing de pilha de chamadas ao invocar APIs do Windows que são mais propensas a serem monitoradas. Por exemplo, ele chama diretamente RtlInitUnicodeString, uma API do Windows que apenas realiza manipulação de strings, em vez de usar o spoofing de pilha.

A técnica mencionada acima pode ser observada nas figuras abaixo. Na primeira figura, podemos ver uma pilha de chamadas típica quando o explorer.exe invoca a função CreateFileW. A ferramenta de monitoramento do sistema, SysMon, efetivamente percorre a pilha de chamadas, permitindo-nos entender o propósito por trás dessa chamada de API e examinar os módulos e funções envolvidos no processo.

K	0	FLTMGR.SYS	FltpPerformPreCallbacks + 0x2fd	0xfffff80cb44d555d	C:\Windows\System32\drivers\FLTMGR.SYS
K	1	FLTMGR.SYS	FltpPassThroughInternal + 0x8c	0xfffff80cb44d50bc	C:\Windows\System32\drivers\FLTMGR.SYS
K	2	FLTMGR.SYS	FltpCreate + 0x2e5	0xfffff80cb450d545	C:\Windows\System32\drivers\FLTMGR.SYS
K	3	ntoskrnl.exe	IofCallDriver + 0x59	0xfffff8037c36e189	C:\Windows\system32\ntoskrnl.exe
K	4	ntoskrnl.exe	IoCallDriverWithTracing + 0x34	0xfffff8037c3151f4	C:\Windows\system32\ntoskrnl.exe
K	5	ntoskrnl.exe	IopParseDevice + 0x632	0xfffff8037c7e51a2	C:\Windows\system32\ntoskrnl.exe
K	6	ntoskrnl.exe	ObpLookupObjectName + 0x719	0xfffff8037c85c029	C:\Windows\system32\ntoskrnl.exe
K	7	ntoskrnl.exe	ObOpenObjectByNameEx + 0x1df	0xfffff8037c85a62f	C:\Windows\system32\ntoskrnl.exe
K	8	ntoskrnl.exe	IopCreateFile + 0x404	0xfffff8037c7c0874	C:\Windows\system32\ntoskrnl.exe
K	9	ntoskrnl.exe	NtCreateFile + 0x79	0xfffff8037c7c0459	C:\Windows\system32\ntoskrnl.exe
K	10	ntoskrnl.exe	KiSystemServiceCopyEnd + 0x25	0xfffff8037c475085	C:\Windows\system32\ntoskrnl.exe
U	11	ntdll.dll	NtCreateFile + 0x14	0x7ffefc8df034	C:\Windows\SYSTEM32\ntdll.dll
U	12	KERNELBASE.dll	CreateFileInternal + 0x2f6	0x7ffef8a8fb26	C:\Windows\System32\KERNELBASE.dll
U	13	KERNELBASE.dll	CreateFileW + 0x66	0x7ffef8a8f816	C:\Windows\System32\KERNELBASE.dll
U	14	windows.storage.dll	CCachedINIFile::Load + 0x59	0x7ffef941ad49	C:\Windows\System32\windows.storage.dll
U	15	windows.storage.dll	CPrivateProfileCache::_AddNewINIFromFile + 0x67	0x7ffef941ac1b	C:\Windows\System32\windows.storage.dll
U	16	windows.storage.dll	CPrivateProfile::Initialize + 0x3bd	0x7ffef9443c7d	C:\Windows\System32\windows.storage.dll
U	17	windows.storage.dll	SHGetCachedPrivateProfile + 0x6e	0x7ffef94839f6	C:\Windows\System32\windows.storage.dll
U	18	windows.storage.dll	CFSFolder::_GetDesktopIni + 0x73	0x7ffef94838bb	C:\Windows\System32\windows.storage.dll
U	19	windows.storage.dll	CFSFolder::_DiscoverLocalizedName + 0x5a9	0x7ffef943b2a9	C:\Windows\System32\windows.storage.dll
U	20	windows.storage.dll	CFSFolder::_CreateIDList + 0x130	0x7ffef943a5d0	C:\Windows\System32\windows.storage.dll
U	21	windows.storage.dll	CFSFolder::ParseDisplayName + 0x911	0x7ffef9438be1	C:\Windows\System32\windows.storage.dll
U	22	shlwapi.dll	IShellFolder_ParseDisplayName + 0x76	0x7ffef9a97886	C:\Windows\System32\shlwapi.dll
U	23	explorerframe.dll	GetRealIDL + 0x107	0x7ffee11394af	C:\Windows\system32\explorerframe.dll
U	24	explorerframe.dll	SimpleToRealIDListWithContext + 0x9b	0x7ffee1139997	C:\Windows\system32\explorerframe.dll
U	25	explorerframe.dll	CNscChangeNotifyTask::_ConvertIDList + 0x17d	0x7ffee10c7a7d	C:\Windows\system32\explorerframe.dll
U	26	explorerframe.dll	CNscChangeNotifyTask::InternalResumeRT + 0x19	0x7ffee10c71a9	C:\Windows\system32\explorerframe.dll
U	27	explorerframe.dll	CRunnableTask::Run + 0xb2	0x7ffee0ff70c2	C:\Windows\system32\explorerframe.dll
U	28	windows.storage.dll	CShellTask::TT_Run + 0x3c	0x7ffef94ab3ec	C:\Windows\System32\windows.storage.dll
U	29	windows.storage.dll	CShellTaskThread::ThreadProc + 0xdd	0x7ffef94ab0a5	C:\Windows\System32\windows.storage.dll
U	30	windows.storage.dll	CShellTaskThread::s_ThreadProc + 0x35	0x7ffef94aaf85	C:\Windows\System32\windows.storage.dll
U	31	shcore.dll	ExecuteWorkItemThreadProc + 0x16	0x7ffef9d52ac6	C:\Windows\System32\shcore.dll
U	32	ntdll.dll	RtlpTpWorkCallback + 0x165	0x7ffefc89c4d5	C:\Windows\SYSTEM32\ntdll.dll
U	33	ntdll.dll	TppWorkerThread + 0x644	0x7ffefc85bec4	C:\Windows\SYSTEM32\ntdll.dll
U	34	KERNEL32.DLL	BaseThreadInitThunk + 0x14	0x7ffefbe27e94	C:\Windows\System32\KERNEL32.DLL
U	35	ntdll.dll	RtlUserThreadStart + 0x21	0x7ffefc8a7ad1	C:\Windows\SYSTEM32\ntdll.dll

Figura 3 –Rastreamento de pilha a partir da explorer.exe chamada CreateFileW.

Em contraste, a próxima figura mostra a pilha de chamadas registrada pelo SysMon quando o DodgeBox usa spoofing de pilha para chamar a função CreateFileW. Observe que não há indicação dos módulos do DodgeBox que acionaram a chamada da API. Em vez disso, todos os módulos envolvidos parecem ser módulos legítimos do Windows.



Frame	Module	Location	Address	Path
K 0	FLTMGR.SYS	FltpPerformPreCallbacks + 0x2fd	0xfffff80cb44d555d	C:\Windows\System32\drivers\FLTMGR.SYS
K 1	FLTMGR.SYS	FltpPassThroughInternal + 0x8c	0xfffff80cb44d50bc	C:\Windows\System32\drivers\FLTMGR.SYS
K 2	FLTMGR.SYS	FltpCreate + 0x2e5	0xfffff80cb450d545	C:\Windows\System32\drivers\FLTMGR.SYS
K 3	ntoskrnl.exe	IofCallDriver + 0x59	0xfffff8037c36e189	C:\Windows\system32\ntoskrnl.exe
K 4	ntoskrnl.exe	IoCallDriverWithTracing + 0x34	0xfffff8037c3151f4	C:\Windows\system32\ntoskrnl.exe
K 5	ntoskrnl.exe	IopParseDevice + 0x632	0xfffff8037c7e51a2	C:\Windows\system32\ntoskrnl.exe
K 6	ntoskrnl.exe	ObpLookupObjectName + 0x719	0xfffff8037c85c029	C:\Windows\system32\ntoskrnl.exe
K 7	ntoskrnl.exe	ObOpenObjectByNameEx + 0x1df	0xfffff8037c85a62f	C:\Windows\system32\ntoskrnl.exe
K 8	ntoskrnl.exe	IopCreateFile + 0x404	0xfffff8037c7c0874	C:\Windows\system32\ntoskrnl.exe
K 9	ntoskrnl.exe	NtCreateFile + 0x79	0xfffff8037c7c0459	C:\Windows\system32\ntoskrnl.exe
K 10	ntoskrnl.exe	KiSystemServiceCopyEnd + 0x25	0xfffff8037c475085	C:\Windows\system32\ntoskrnl.exe
U 11	ntdll.dll	NtCreateFile + 0x14	0x7ffefc8df034	C:\Windows\System32\ntdll.dll
U 12	KernelBase.dll	ARI::DependencyMiniRepository::LogDMRSectionNotFound + 0x7c	0x7ffef8b3ca3c	C:\Windows\System32\KernelBase.dll
U 13	kernel32.dll	BaseThreadInitThunk + 0x14	0x7ffefbe27e94	C:\Windows\System32\kernel32.dll
U 14	ntdll.dll	RtlUserThreadStart + 0x21	0x7ffefc8a7ad1	C:\Windows\System32\ntdll.dll

Figura 4 – Rastreamento de pilha de chamada do DodgeBox.

Quando a função CallFunction é invocada, o DodgeBox usa um gadget aleatório jmp qword ptr [rbp+48h] que reside na seção .text de KernelBase. O DodgeBox analisa os códigos de desenrolamento dentro da seção .pdata para extrair o tamanho do desenrolamento para a função que inclui o gadget selecionado. O DodgeBox obtém os endereços de RtlUserThreadStart + 0x21e e BaseThreadInitThunk + 0x14, juntamente com seus respectivos tamanhos de desenrolamento. O DodgeBox configura a pilha inserindo os endereços de RtlUserThreadStart + 0x21, BaseThreadInitThunk + 0x14 e o endereço do gadget nas posições corretas, utilizando os tamanhos de desenrolamento recuperados. Depois disso, o DodgeBox prossegue para inserir o endereço de retorno apropriado em [rbp+48h] e prepara os registradores e a pilha com os valores de argumento necessários para serem passados para a API. Essa preparação garante que a API seja chamada corretamente e com os parâmetros pretendidos. Por fim, o DodgeBox executa uma instrução jmp para redirecionar o fluxo de controle para a API de destino.

3 MITRE ATT&CK - TTPs

Tática	Técnica	Detalhes
Defense Evasion	T1480 T1480.001 T1027 T1027.007 T1620 T1562.001	Consiste em técnicas que os adversários usam para evitar a detecção durante seu comprometimento.
Execution	T1106	Consiste em técnicas que resultam em código controlado pelo adversário em execução em um sistema local ou remoto.
Persistence	T1574.002	Consiste em técnicas que os adversários usam para manter o acesso aos sistemas em reinicializações, credenciais alteradas e outras interrupções que podem cortar seu acesso.

Tabela 1 – Tabela MITRE ATT&CK.

4 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:

Use autenticação multifator

- Além da senha, é necessária outra forma de confirmação, como o envio de uma mensagem de texto para o seu celular.

Escolha senhas mais difíceis de serem descobertas

- Uma senha forte terá pelo menos 12 caracteres e incluirá uma mistura de letras maiúsculas e minúsculas, números e símbolos.

Mantenha os softwares atualizados

- Isso inclui o sistema operacional e todos os aplicativos que você usa.

Gerencie as configurações de mídias sociais

- Mantenha a maior parte das informações pessoais e privadas bloqueadas.

Proteja a rede wireless

- Use uma senha que utiliza criptografia forte.

Instale e mantenha atualizado um antivírus ou firewall

- Isso pode ajudar a proteger seu dispositivo contra ameaças.

5 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:	0d068b6d0523f069d1ada59c12891c4a
sha1:	9ad85457947b5ba0efea57fcb2df0653ac70c3f8
sha256:	3a7dfc0850136c59104d362b11183a5a61511d056ef393f6a6a63fdba9bbb804
File name:	Music.zip

Indicadores de compromisso do artefato	
md5:	b3067f382d70705d4c8f6977a7d7bee4
sha1:	1d739e433f8b8258ea4378665822b6b6c652a607
sha256:	bc92e8e964e0492b3595d9470e59941bded90082040ac436583b9f3269e1e550
File name:	SandboxieWUAU.exe

Indicadores de compromisso do artefato	
md5:	d72f202c1d684c9a19f075290a60920f
sha1:	2cc76a0434a1d489c1547c7021a3dd68499141c3
sha256:	c6a3a1ea84251aed908702a1f2a565496d583239c5f467f5dcd0cfc5bfb1a6db
File name:	sbiedll.dll

Indicadores de compromisso do artefato	
md5:	d72f202c1d684c9a19f075290a60920f
sha1:	2cc76a0434a1d489c1547c7021a3dd68499141c3
sha256:	c6a3a1ea84251aed908702a1f2a565496d583239c5f467f5dcd0cfc5bfb1a6db
File name:	sbiedll.dll

Indicadores de compromisso do artefato	
md5:	d72f202c1d684c9a19f075290a60920f
sha1:	2cc76a0434a1d489c1547c7021a3dd68499141c3
sha256:	c6a3a1ea84251aed908702a1f2a565496d583239c5f467f5dcd0cfc5bfb1a6db
File name:	sbiedll.dll

Indicadores de compromisso do artefato	
md5:	294cc02db5a122e3a1bc4f07997956da
sha1:	cbe737dc5f427d0d6202132e859ae25b4f48574c
sha256:	ed606d718874c29b9a1101775069d694b67eb5a4492404ddd98ebfcdcfcce205
File name:	sbiedll.dat

Indicadores de compromisso do artefato	
md5:	393065ef9754e3f39b24b2d1051eab61
sha1:	c3874d5cc7e82ad373b67a3650b0dfee7c219f8f
sha256:	33fd050760e251ab932e5ca4311b494ef72cee157b20537ce773420845302e49
File name:	22.exe

Tabela 2 – Indicadores de Compromissos de artefatos

6 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- **ZScaler**
- **Thehackernews**

7 AUTORES

- **Leonardo Oliveira Silva**

heimdall
security research
A DIVISION OF ISH