BOLETIM DE SEGURANÇA

Grupo Lazarus e Rootkit do BYOVD ao Zero Day

heimdall
security research
A DIVISION OF ISH

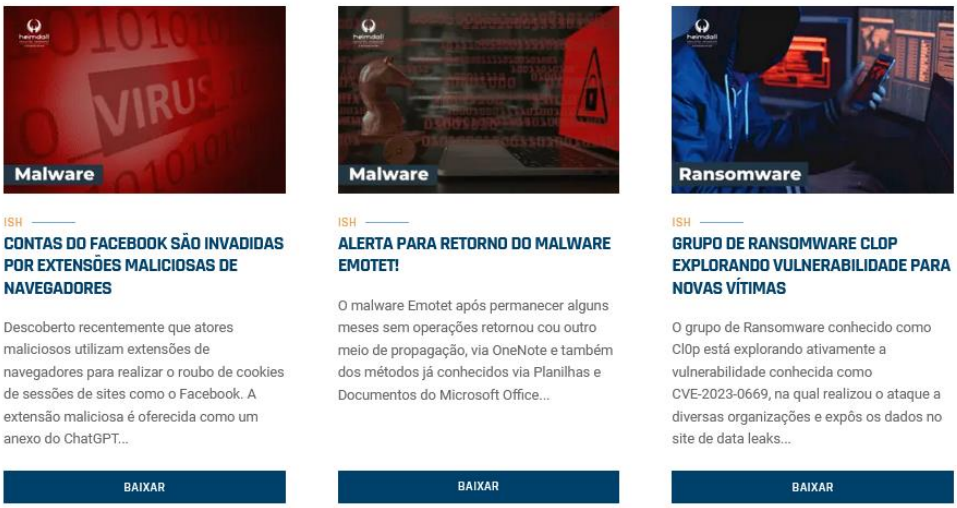
TLP: CLEAR

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

A equipe de pesquisadores da AVAST descobriu uma exploração de administrador para kernel, uma vulnerabilidade zero day anteriormente desconhecida no driver AppLocker appid.sys em sistemas Windows. A Microsoft corrigiu esta vulnerabilidade como CVE-2024-21338 na atualização Patch Tuesday de fevereiro. Essa atividade de exploração foi orquestrada pelo notório Grupo Lazarus, com o objetivo final de estabelecer uma primitiva de leitura/gravação do kernel. A princípio, isso permite o grupo realize manipulação direta de objetos do kernel em uma versão atualizada de seu rootkit FudModule somente para dados, cuja versão anterior foi analisada pela ESET e AhnLab.

2 INFORMAÇÕES SOBRE A AMEAÇA

Após a realização de engenharia reversa completa dessa variante de rootkit atualizada, identificou-se avanços substanciais em termos de funcionalidade e furtividade, com quatro técnicas de rootkit novas – e três atualizadas. Em um avanço importante, o rootkit agora emprega uma nova técnica de manipulação na entrada de tabela de identificador em uma tentativa de suspender processos protegidos como o *Protected Process Light* (**PPL**) associados ao Microsoft Defender, **CrowdStrike Falcon** e **HitmanPro**. Outro avanço significativo é a exploração da vulnerabilidade de dia zero, onde o **Lazarus** utilizou anteriormente técnicas BYOVD (Bring Your Own Vulnerable Driver) muito mais barulhentas para cruzar a fronteira do administrador para o kernel. Na análise também foi possível recuperar grandes partes da cadeia de infecção que levou à implantação do rootkit, resultando na descoberta de um novo RAT (Trojan de acesso remoto) atribuído ao Lazarus.

3 ANÁLISE DA EXPLORAÇÃO

Os invasores com privilégios de administrador frequentemente obtêm acesso no nível do kernel explorando drivers vulneráveis conhecidos, em uma técnica chamada *Bring Your Own Vulnerable Driver* (**BYOVD**). O invasor, pode passar do administrador para o kernel abrindo um novo campo de possibilidades. Com o acesso no nível do kernel, um invasor pode interromper o software de segurança, ocultando indicadores de infecção (incluindo arquivos, atividade de rede, processos etc.), desativando a telemetria do modo kernel, mitigações e muito mais. Além disso, como a segurança do *Protected Process Light* (PPL) depende do limite do administrador ao kernel, o invasor também ganha a capacidade de adulterar processos protegidos ou adicionar proteção a um processo arbitrário. Isso pode ser especialmente poderoso se o **lsass** estiver protegido com RunAsPPL, pois ignora o PPL podendo permitir que o ator malicioso despeje credenciais que de outra forma seriam inacessíveis.

O ator de ameaça pode aproveitar o BYOVD para explorar uma vulnerabilidade de *N DAY* conhecida publicamente. Isso é muito fácil de realizar, pois há muitas explorações públicas de prova de conceito (**POC**) para várias vulnerabilidades. Entretanto, é relativamente simples de detectar, já que o invasor deve primeiro colocar um driver vulnerável conhecido no sistema de arquivos e depois carregá-lo no kernel, resultando em duas grandes oportunidades de detecção. Em alguns sistemas podem ter a lista de bloqueio de drivers vulneráveis da Microsoft habilitada, o que bloquearia o carregamento de alguns dos drivers vulneráveis mais comuns. Versões anteriores do **rootkit FudModule** poderiam ser colocadas nesta categoria, explorando inicialmente uma vulnerabilidade conhecida em dbutil_2_3.sys e depois passando para o ene.sys em versões posteriores.

Já em cenários mais sofisticados, um invasor usaria BYOVD para explorar uma vulnerabilidade zero day em um driver assinado de terceiros. Isso exige que o invasor descubra primeiro essa vulnerabilidade, o que inicialmente pode parecer uma tarefa difícil. No entanto, qualquer vulnerabilidade explorável em qualquer driver assinado servirá e, infelizmente, não faltam drivers de terceiros de baixa qualidade. Portanto, o nível de dificuldade de descobrir tal vulnerabilidade pode não ser tão elevado como poderia parecer inicialmente, podendo ser suficientemente necessário verificar uma coleção de drivers em busca de padrões de vulnerabilidade conhecidos, de acordo com os pesquisadores da Carbon Black que recentemente usaram análise estática em massa para descobrir 34 vulnerabilidades exclusivas em mais de 200 drivers assinados. Esses ataques BYOVD zero day são notavelmente mais furtivos do que os ataques de n dias, uma vez que os analistas de segurança não podem mais confiar em hashes de drivers vulneráveis conhecidos para detecção.

O exploit do grupo Lazarus começa com um estágio de inicialização, executando uma configuração única tanto para o exploit quanto para o rootkit (ambos foram compilados no mesmo módulo). Na inicialização, começa resolvendo dinamicamente todas as funções necessárias da API do Windows, seguida por uma verificação de antidepuração de baixo esforço no **PEB.BeingDebugged**. Em seguida, a exploração inspeciona o número da compilação para ver se está sendo executado em uma versão compatível do Windows. Nesse caso, é carregada constantes codificadas adaptadas à construção atual. A escolha de constantes às vezes se resume à revisão de compilação de atualização (UBR), demonstrando um alto grau de dedicação para garantir que o código seja executado de forma limpa em uma ampla variedade de máquinas alvo.

```
if ( build_num == 17763 )
{
  cfg->offset_Token = 0x358i64;
  cfg->offset_ObjectTable = 0x418i64;
  cfg->offset_ActiveProcessLinks = 0x2E8i64;
  cfg->offset_ImageFileName = 0x450i64;
  cfg->offset_MitigationFlags = 0x820i64;
  cfg->offset_Protection = 0x6CAi64;
  cfg->offset_EtwpActiveSystemLoggers = 0x1080i64;
  cfg->offset_EtwpGuidHashTable = 0x1B8i64;
  cfg->offset_ProviderEnableInfo = 0x50i64;
  syscall_NtSuspendThread = 0x1B5;
  syscall_NtSuspendProcess = 0x1B4;
  syscall_NtResumeProcess = 0x174;
}
```

Figura 1 – Trecho de código decodificado.

No processo de inicialização, os endereços base de três módulos do kernel, **ntoskrnl**, **netio**, e **fltmgr**, continuam vazando, chamando **NtQuerySystemInformation** usando a classe **SystemModuleInformation**. O endereço **KTHREAD** do thread atualmente em execução também é vazado de maneira semelhante, duplicando o pseudo handle do thread atual e, em seguida, encontrando o endereço do objeto do kernel correspondente usando a classe **SystemExtendedHandleInformation** de informações do sistema. Finalmente, a exploração carrega manualmente a imagem **ntoskrnl** no espaço de endereço do usuário, apenas para procurar endereços virtuais relativos (**RVAs**) de algumas funções de interesse.

Como o driver **appid.sys** não precisa estar carregado na máquina de destino, o exploit pode primeiro ter que carregá-lo. Ele opta por fazer isso de maneira indireta, gravando um evento em um provedor específico de Event Tracing for Windows (**ETW**) relacionado ao AppLocker. Depois que o appid.sysde for carregado, o exploit representa a conta local service usando um syscall direto para a classe NtSetInformationThread com a ThreadImpersonationToken de informações do thread. Ao personificar local service, agora ele pode obter um identificador de *leitura/gravação* para *\Device\AppId*. Com esse identificador, a exploração finalmente prepara o buffer de entrada IOCTL e aciona a vulnerabilidade usando o NtDeviceIoControlFilesyscall.

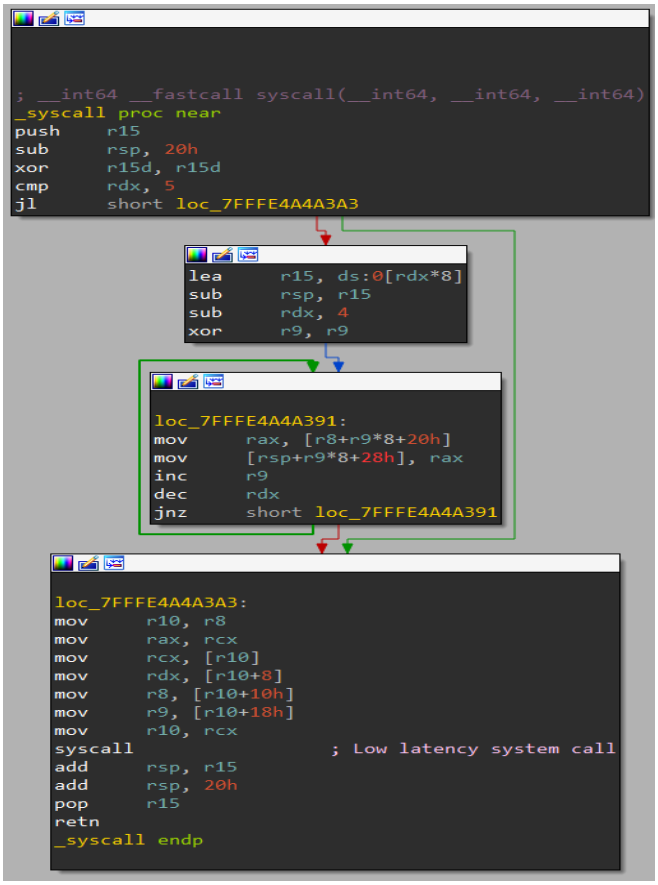


Figura 2 – Syscalls durante a exploração.

A exploração cria um buffer de entrada **IOCTL** em que o retorno de chamada vulnerável é essencialmente um gadget executando uma cópia de 64 bits do buffer de entrada IOCTL para um endereço de destino arbitrário. Este endereço foi escolhido para corromper o **PreviousMode** do thread atual. Garantindo que o byte de origem correspondente no buffer de entrada IOCTL seja zero, a cópia limpará o campo PreviousMode, resultando efetivamente na interpretação de seu valor como **KernelMode**. Um direcionamento como PreviousMode é uma técnica de exploração amplamente popular, pois corromper esse byte na estrutura **KTHREAD** ignora as verificações do modo kernel dentro de syscalls como **NtReadVirtualMemory** ou **NtWriteVirtualMemory**, permitindo que um invasor no modo de usuário leia e grave memória arbitrária do kernel.

Observe que, embora essa técnica tenha sido atenuada em algumas compilações do Windows Insider, essa atenuação ainda não atingiu a disponibilidade geral no momento da redação deste artigo. A exploração pode tentar acionar o IOCTL vulnerável duas vezes. Isso se deve a um argumento extra adicionado no **Win11 22H2**. Como resultado, o manipulador IOCTL em compilações mais recentes espera que o buffer de entrada tenha o tamanho de **0x20** bytes, enquanto, anteriormente, o tamanho esperado era de apenas **0x18** bytes. Em vez de selecionar o tamanho do buffer de entrada adequado para a compilação atual, a exploração apenas tenta chamar o IOCTL duas vezes: primeiro com um tamanho de buffer de entrada 0x18 bytes e depois caso não seja bem-sucedido – com 0x20 bytes. Esta é uma abordagem válida, pois a primeira ação do IOCTL é verificar o tamanho do buffer de entrada e, se não corresponder ao tamanho esperado, retornará imediatamente **STATUS_INVALID_PARAMETER**.

Para verificar se foi bem-sucedido, o exploit emprega o syscall NtWriteVirtualMemory, tentando ler o thread atual PreviousMode(o Lazarus evita usar NtReadVirtualMemory). Se a exploração for bem-sucedida, o syscall deverá retornar STATUS_SUCCESS e o PreviousModebyte vazado deverá ser igual a 0 (significando KernelMode). Caso contrário, o syscall deverá retornar um código de status de erro, pois seria impossível ler a memória do kernel sem um arquivo PreviousMode.

Se tratando de exploração do administrador para o kernel o objetivo era corromper o arquivo PreviousMode. Permitindo uma forte primitiva de leitura/gravação do kernel, onde o thread de modo de usuário afetado pode ler e gravar memória arbitrária do kernel usando a syscalls **Nt(Read|Write)VirtualMemory**. O rootkit **FudModule** emprega técnicas de manipulação direta de objetos do kernel (**DKOM**) para interromper vários mecanismos de segurança do kernel. Vale a pena reforçar que o FudModule é um rootkit somente de dados, o que significa que ele é executado inteiramente a partir do espaço do usuário e toda a adulteração do kernel é realizada através de leitura/gravação.

As análises detalhadas das variantes do rootkit FudModule foram descobertas em setembro de 2022 de forma independente pelas equipes de pesquisa da AhnLab e da ESET, recebendo o nome da FudModule.dllstring usada como nome em sua tabela de exportação. Embora este artefato não esteja mais presente, não há dúvida que foram encontrados, se trata de uma versão atualizada do mesmo rootkit. Essa descoberta diz respeito a uma amostra que apresenta nove técnicas de rootkit e que explora uma vulnerabilidade anteriormente desconhecida do administrador para o kernel. Destas nove técnicas, quatro são novas, três são melhoradas e duas permanecem inalteradas em relação às variantes anteriores. Isso deixa duas das sete técnicas originais, que foram obsoletas e não estão mais presentes na variante mais recente.

Cada técnica utilizada pelo rootkit recebe um bit, variando de **0x1** a **0x200** (o bit **0x20** não é utilizado na variante atual). O FudModule executa as técnicas sequencialmente, em ordem crescente dos bits atribuídos. Os bits são usados para relatar o sucesso das técnicas individuais. Durante a execução, o FudModule construirá um valor inteiro (nomeado bitfield_techniques), onde serão definidos apenas os bits correspondentes às técnicas executadas com sucesso. Esse número inteiro é finalmente gravado em um arquivo chamado tem1245.tmp, informando sobre o sucesso do rootkit. Não foram encontrados esse nome de arquivo referenciado em nenhuma outra amostra do Lazarus, sugerindo que o arquivo descartado só é inspecionado por meio de atividade prática no teclado, presumidamente por meio de um comando Trojan de acesso remoto (**RAT**). Isso mostra que a crenças de que o FudModule está apenas vagamente integrado ao restante do ecossistema de malware do Lazarus e que o Lazarus é muito cuidadoso ao usar o rootkit, implantando-o apenas sob demanda nas circunstâncias certas.

```
context = (__int64 *)LocalAlloc(0x40u, 0x1C0ui64);
context[51] = a1;
context[52] = a2;
result = setup(context);
if ( !(_DWORD)result )
{
  result = exploit(context);
  if ( !(_DWORD)result )
  {
    bitfield_techniques = registry_callbacks(context) != 0;
    if ( (unsigned int)object_callbacks(context) )
      bitfield_techniques |= 2u;
    if ( (unsigned int)process_image_thread_callbacks(context) )
      bitfield_techniques |= 4u;
    if ( (unsigned int)minifilters(context) )
      bitfield_techniques |= 8u;
    if ( (unsigned int)wfp_callouts(context) )
      bitfield_techniques |= 0x10u;
    if ( (unsigned int)etw_system_loggers(context) )
      bitfield_techniques |= 0x40u;
    if ( (unsigned int)etw_provider_guids(context) )
      bitfield_techniques |= 0x80u;
    if ( (unsigned int)image_verification_callbacks(context) )
      bitfield_techniques |= 0x100u;
    if ( (unsigned int)direct_attacks((__int64)context) )
      bitfield_techniques |= 0x200u;
    restore_previousmode((__int64)context);
    memset(context, 0, 0x1C0ui64);
    LocalFree(context);
```

Figura 3 – Função principal do rootkit

4 CONCLUSÃO

O grupo Lazarus se mantém como um dos principais e mais ativos e persistentes agentes em termos de ameaças cibernéticas avançadas. Suas estratégias e métodos já são amplamente conhecidos, mas ainda assim, eles conseguem novos níveis de sofisticação técnica de tempos em tempos. Um exemplo disso é o rootkit FudModule, que se destaca como uma das ferramentas mais elaboradas no inventário do Lazarus.

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 URL, IPs e Domínios

Indicadores de URL, IPs e Domínios	
URL	https://github.com/avast/ioc/tree/master/FudModule#yara

Tabela 1 – Indicadores de Compromissos de Rede.

6 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:

Soluções antivírus/anti-malware

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

Firewalls

- Ative firewalls no nível do sistema operacional e da rede para filtrar tráfego não autorizado que pode ser usado para transmitir dados maliciosos ou comandos de controle.

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.

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 e-mail.

Monitoramento de rede

- Use ferramentas de monitoramento de rede para detectar atividades suspeitas, como padrões incomuns de tráfego ou comunicações com endereços IP maliciosos conhecidos.

Backup de dados

- Realize backups regulares de dados importantes e garanta que esses backups estejam seguros e possam ser restaurados. Isso minimiza o dano em caso de infecção por malware.

Segurança de e-mail

- Implemente soluções de segurança de e-mail para filtrar mensagens de phishing e anexos maliciosos. Isso reduz a probabilidade de infecção por malware através de e-mails.

Isolamento de sistemas sensíveis

- Isolar sistemas e redes sensíveis de outras partes da infraestrutura de TI pode ajudar a limitar a propagação de malwares.

Análise e resposta a incidentes

- Tenha um plano de resposta a incidentes para lidar rapidamente com qualquer suspeita de infecção por malware. Isso deve incluir procedimentos para isolar sistemas infectados, analisar o ataque e restaurar os sistemas a partir de backups limpos.

7 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- **DecodedAvast**
- **Microsoft**

heimdall
security research
A DIVISION OF ISH