TLP: CLEAR

BOLETIM DE SEGURANÇA

Novo Rootkit furtivo Pumakit em sistemas Linux

heimdall
security research

Acesse a nossa nova comunidade através do WhatsApp!

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 INFORMAÇÕES SOBRE A AMEAÇA

Recentemente a Elastic identificou um malware denominado Pumakit em um arquivo binário suspeito ('cron'). No entanto, não há informações sobre quem está utilizando o malware ou quais são seus alvos. O PUMAKIT é um malware avançado, detectado durante uma análise de ameaças no VirusTotal. Ele foi nomeado com base em strings encontradas dentro de seu binário pelo desenvolvedor. Sua estrutura é composta por várias etapas, incluindo um dropper (cron), dois executáveis residentes na memória (/memfd:tgte e /memfd:wpn), um módulo rootkit LKM e um rootkit userland em forma de objeto compartilhado (SO).

As funcionalidades essenciais do módulo do kernel abrangem a elevação de privilégios, a ocultação de arquivos e diretórios, a dissimulação de ferramentas do sistema, a implementação de técnicas antidepuração e a comunicação com servidores de comando e controle (C2). Foi encontrado um binário intrigante chamado cron. Esse binário foi carregado pela primeira vez em setembro de 2024, sem nenhuma detecção, o que levantou suspeitas sobre sua possível furtividade. Após uma análise, descobriu-se outro artefato relacionado, /memfd:wpn (deleted)71cc6a6547b5afda1844792ace7d5437d7e8d6db1ba995e1b2fb760699693f24, também carregado no mesmo dia e igualmente sem detecções.

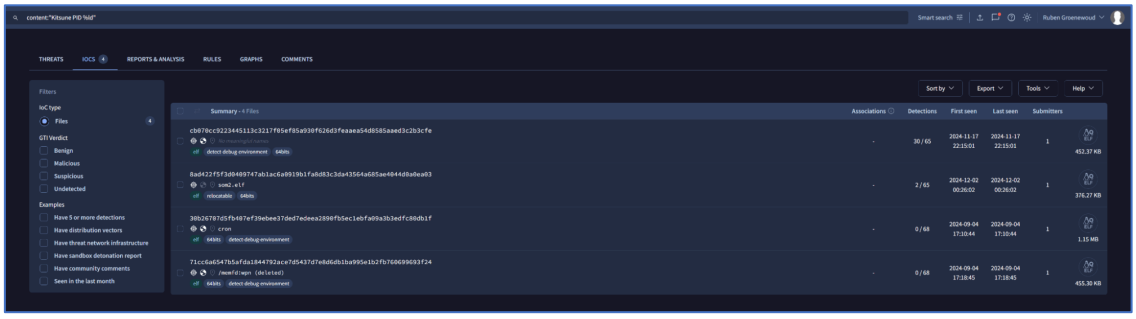

Figura 1 – Realização de hunting no virustotal.

O PUMAKIT, batizado em referência ao seu módulo rootkit LKM integrado (denominado "PUMA" pelos criadores do malware) e ao rootkit userland Kitsune, utiliza uma arquitetura em várias etapas, começando com um dropper que dá início a uma cadeia de execução. O processo se inicia com o cron binário, que gera dois executáveis residentes na memória: /memfd:tgt (deleted) e /memfd:wpn (deleted). Enquanto o /memfd:tgt funciona como um binário Cron inofensivo, o /memfd:wpn atua como um carregador de rootkit. Este carregador é responsável por avaliar as condições do sistema, executar um script temporário (/tmp/script.sh) e, por fim, implantar o rootkit LKM. O rootkit LKM inclui um arquivo SO embutido - Kitsune - para interagir com o rootkit no espaço do usuário.

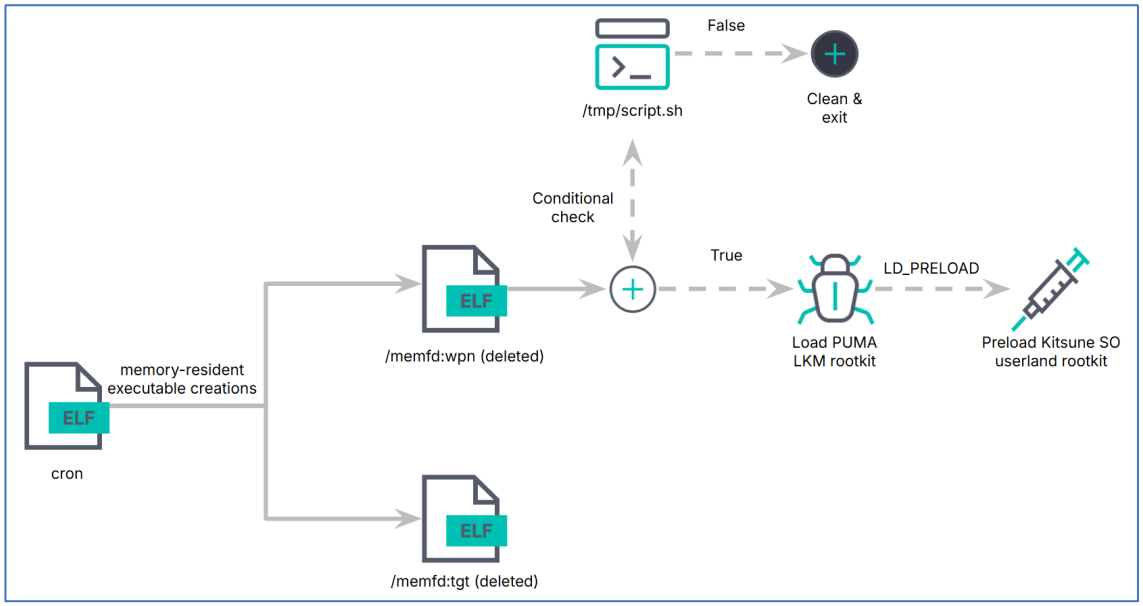

Figura 2 – Cadeia de infecção PUMAKIT.

O cronbinário funciona como um dropper. A função a seguir é o principal controlador lógico em uma amostra do malware PUMAKIT. Em essência, o malware busca manter-se discreto. Quando executado normalmente (sem um argumento específico), ele roda binários ELF ocultos sem deixar vestígios no disco, possivelmente disfarçando-se como um processo legítimo (como o cron).

```
004014f0    int64_t mw_main(int32_t argc, char**& argv)

00401502        char** argv_1
00401502        char* rbp_1 = &argv_1[1]
00401509        char* argv_ = *argv_1
0040150c        char* strHuinder_1 = strHuinder
0040150c
00401516        while (true)
00401516            if (argv_ == 0)
00401567                int32_t fd = writeToMemfd(name: "tgt", Elf: tgtElfp, size: tgtSize)
00401567
00401570                if (fd s>= 0)
0040158b                    int32_t wpnfd = writeToMemfd(name: "wpn", Elf: wpnElfp, size: wpnSize)
0040158b
00401594                    if (wpnfd s>= 0)
0040159a                        pid_t childPid
0040159a                        struct rusage* rusage
0040159a                        childPid, rusage = fork()
0040159a
004015a3                        if (childPid s>= 0)
004015a5                            char*** envp
004015a5                            int32_t flags
004015a5
004015a5                            if (childPid == 0)
004015c0                                int32_t fdDevNull
004015c0                                int32_t flags_1
004015c0                                fdDevNull, flags_1 = openat(dirfd: devNull, pathname: 1)
004015c0
004015ca                                if (fdDevNull s>= 0 && dup2(oldfd: fdDevNull, newfd: 1) s>= 0)
004015e4                                    int32_t newFd
004015e4                                    char*** envp_1
004015e4                                    newFd, envp_1 = dup2(oldfd: fdDevNull, newfd: 2)
004015e4
004015eb                                    if (newFd s>= 0)
004015f9                                        envp, flags = execveat(dirfd: wpnfd, pathname: &cmdUsrBinSshd, argv: argv_1, envp: envp_1, flags: flags_1)
004015a5                            else
004015ab                                envp, flags = wait4(pid: childPid, wstatus: nullptr, options: 0, rusage)
0040160a                                execveat(dirfd: fd, pathname: argv, argv: argv_1, envp, flags)
0040160a
00401611                            if (wpnfd != 0)
00401615                                close(wpnfd)
00401615
0040161c                            if (fd != 0)
00401620                                close(fd)
00401620
00401632                        return 0
00401632
0040151b                    strlen(s: strHuinder_1)
0040151b
00401530                    if (strcmp(s1: argv_, s2: strHuinder_1) == 0)
00401530                        break
00401530
00401548                    argv_ = *rbp_1
0040154c                    rbp_1 = &rbp_1[8]
0040154c
004018c4            if (argc s<= 2)
004019a5                return 42
004019a5
```

Figura 3 – Principal função do conta-gotas inicial.

Ao verificar o arquivo ELF /memfd:tgt, percebe-se que ele é o binário padrão do Cron do Ubuntu Linux, sem modificações. Já o arquivo /memfd:wpn é mais intrigante, pois é o binário encarregado de carregar o rootkit LKM. Este carregador de rootkit tenta se ocultar imitando o executável /usr/sbin/sshd. Ele verifica certos pré-requisitos, como a habilitação do secure boot e a disponibilidade dos símbolos necessários. Se todas as condições forem satisfeitas, ele carrega o rootkit do módulo do kernel. No Kibana, nota-se que o programa verifica a habilitação do secure boot consultando o dmesg. Se as condições corretas forem encontradas, um script de shell chamado script.sh é criado no diretório /tmp e executado.

process.executable	process.command_line	process.parent.executable	process.parent.command_line	file.path	event.action
/bin/sh	sh -c bash /tmp/script.sh "/boot/vmlinuz-5.10.0-33-cloud-amd64"	/dev/fd/4	/usr/bin/sshd -t	-	exec
/dev/fd/4	-	-	-	/tmp/script.sh	creation
/bin/sh	sh -c dmesg | grep 'ecure boot enabled'	/dev/fd/4	/usr/bin/sshd -t	-	exec
/dev/fd/4	/usr/bin/sshd -t	./30b26707d5fb407ef39ebee37ded7edeea2890fb5ec1ebfa09a3b3edfc80db1f	./30b26707d5fb407ef39ebee37ded7edeea2890fb5ec1ebfa09a3b3edfc80db1f	-	exec
./30b26707d5fb407ef39ebee37ded7edeea2890fb5ec1ebfa09a3b3edfc80db1f	./30b26707d5fb407ef39ebee37ded7edeea2890fb5ec1ebfa09a3b3edfc80db1f	/bin/bash	bash	-	exec

Figura 4 – Fluxo de execução do script bash e do carregador rootkit começando em /dev/fd/4.

A habilidade do rootkit LKM de modificar o comportamento do sistema começa com o uso da tabela syscall e a dependência de kallsyms_lookup_name() para resolver símbolos. Diferente dos rootkits modernos voltados para versões de kernel 5.7 e superiores, este rootkit não utiliza kprobes, sugerindo que foi desenvolvido para kernels mais antigos.

```
000032e0    int64_t init_module()

000032f9        void* rdi
000032f9        int64_t var_38 = __fentry__(rdi)
000032ff        void* i = nullptr
00003301        sys_call_table = kallsyms_lookup_name(name: "sys_call_table")
00003301
```

Figura 5 – Resolvendo um ponteiro para sys_call_table usando kallsyms_lookup_name.

No rootkit, as funções getdents_hook() e getdents64_hook() são responsáveis por manipular as chamadas de sistema de listagem de diretórios para esconder arquivos e pastas dos usuários. As chamadas de sistema getdents() e getdents64() são utilizadas para ler entradas de diretórios. O rootkit intercepta essas funções para filtrar quaisquer entradas que atendam a critérios específicos. Em particular, arquivos e diretórios com o prefixo zov_ são ocultados de qualquer usuário que tente listar o conteúdo de um diretório.

Foi identificado no rootkit outro arquivo ELF dentro do objeto do kernel. Após extrair esse binário, descobriu-se que se tratava do arquivo /lib64/libs.so. Durante a análise, foram encontradas várias referências a strings como Kitsune PID %ld, sugerindo que os desenvolvedores chamam o SO de Kitsune.

Depois que o dropper é executado, um evento é registrado no syslog. Esse evento indica que um processo foi iniciado com uma pilha executável.

```
[687.108154]
process'/home/ruben_groenewoud/30b26707d5fb407ef39ebee37ded7edeea2890fb5ec1ebfa09a3b3edfc
80db1f' started with executable stack
```

Tabela 1 – Registro de evento no syslog.

O descritor de arquivo continuará sendo o processo pai do dropper até sua conclusão, o que levará à execução de vários arquivos por meio desse processo pai. Após o descarte de /tmp/script.shele, pode-se detectar sua execução verificando a descoberta de atributos de arquivo e a atividade de desarquivamento.

```
process where host.os.type == "linux" and event.type == "start" and event.action == "exec" and
(process.parent.args like "/boot/*" or process.args like "/boot/*") and ( (process.name in ("file", "unlzma",
"gunzip", "unxz", "bunzip2", "unzstd", "unzip", "tar")) or (process.name == "grep" and process.args == "ELF")
or  (process.name in ("lzop", "lz4") and process.args in ("-d", "--decode"))) and not process.parent.name ==
"mkinitramfs"
```

Tabela 2 – Processo em execução.

Após o carregamento do módulo do kernel, é possível observar vestígios da execução do comando pelo processo kthreadd. O rootkit gera novas threads do kernel para executar comandos específicos.

```
cat /dev/null
truncate -s 0 /usr/share/zov_f/zov_latest
```

Tabela 3 – Execução do comando pelo processo kthreadd.

2 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:

Mantenha o sistema e software atualizados

- Certifique-se de que o sistema operacional, drivers e todos os softwares estão sempre atualizados com os patches de segurança mais recentes.

Use software de segurança confiável

- Utilize antivírus e ferramentas de segurança de fornecedores confiáveis que ofereçam proteção contra rootkits.

Habilite a proteção de kernel

- Ative as proteções de kernel disponíveis no seu sistema operacional para dificultar a instalação de rootkits no nível do kernel.

Utilize ferramentas de verificação específicas

- Ferramentas especializadas podem detectar e remover rootkits.

Educação e conscientização

- Treine os usuários para reconhecerem e evitarem táticas de engenharia social, como phishing, que são frequentemente usadas para distribuir rootkits.

Monitoramento contínuo

- Implemente soluções de monitoramento contínuo para detectar atividades suspeitas e anômalas no sistema.

Backup regular

- Realize backups regulares dos dados críticos e mantenha cópias offline para garantir a recuperação em caso de infecção.

3 INDICADORES DE COMPROMETIMENTO (IOC)

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 Comprometimento (IOCs) relacionadas a análise do(s) artefato(s) deste relatório.

Indicadores do artefato	
md5:	4375998ea157a8a21e1ead13052bad8a
sha1:	e0f3e48c7dd577153e4dd46dd13470715f68a5e6
sha256:	cb070cc9223445113c3217f05ef85a930f626d3feaaea54d8585aaed3c2b3cfe
File name:	wpn.elf

Indicadores do artefato	
md5:	b21ae7ada5346dd59f582bba5b19bb31
sha1:	258a88300d071699b463d9791a542235ef233538
sha256:	8ef63f9333104ab293eef5f34701669322f1c07c0e44973d688be39c94986e27
File name:	libs.so

Indicadores do artefato	
md5:	b5793af33aa19112ee45d56c51f268f7
sha1:	a7735259ba8c051d3905dac98c930e26e29e5b5b
sha256:	8ad422f5f3d0409747ab1ac6a0919b1fa8d83c3da43564a685ae4044d0a0ea03
File name:	som2.elf

Indicadores do artefato	
md5:	10913b57d02c52353b3217d1b371e661
sha1:	a46233a2a47c9643c8602606228ba1c03d78588c
sha256:	bbf0fd636195d51fb5f21596d406b92f9e3d05cd85f7cd663221d7d3da8af804
File name:	some1.so

Tabela 4 – Indicadores de Comprometimento

Indicadores de URL, IPs e Domínios

Indicadores de URL, IPs e Domínios	
Domínio	rhel.opsecurity1[.]art sec.opsecurity1.art
IP	89.23.113[.]204

Tabela 5 – Indicadores de Comprometimento 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.

4 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- **Elastic**
- **Bleepingcomputer**

5 AUTORES

- **Leonardo Oliveira Silva**

heimdall
security research
A DIVISION OF ISH