SECURITY INTELLIGENCE

Reduzindo superfície de ataque ao kernel do Linux

Anderson Nascimento
anderson [at] allelesecurity [dot] com [dot] br

Semana da Computação UFBA 2019
SEMCOMP 2019
24/03/2019

ALLELE SECURITY INTELLIGENCE

Agenda

Parte 1

- Sobre
- Objetivo
- Superfície de ataque
- Modelagem de ameaças

Parte 2

- Reduzindo superfície de ataque
- Chamadas de sistemas
- SECCOMP
- BPF
- Módulos de terceiros
- *Namespaces*
- *Capabilities*

Dúvidas?

Sobre

Anderson Nascimento
Co-Fundador e Security Researcher @ Allele Security Intelligence

Twitter: andersonc0d3
Github: andersonc0d3
Linkedin: andersonc0d3
Blog: https://blog.andersonc0d3.io

Anteriormente

Pesquisador em segurança da informação independente

Estuda o kernel do freebsd desde ~ 2010
Estuda o kernel do linux desde ~ 2015

Aviso legal

Este é o resultado de um trabalho pessoal e sem revisões prévias, nascido do interesse do apresentador como uma forma de estudar assuntos de seu interesse, sendo assim, erros poderão ser cometidos nesta apresentação e peço desculpas desde já.

Objetivo

Objetivo desta apresentação é mostrar a superfície de ataque ao *kernel* do Linux e meios para reduzi-las, diminuindo a quantidade de código exposto para um possível atacante.

Não é objetivo desta apresentação mostrar todos nem os mais prováveis vetores utilizados em ataques ao *kernel* do Linux. Nesta apresentação somente os vetores de ataque que influenciou de alguma forma a pesquisa realizada por este apresentador serão discutidos.

Superfície de ataque

A superfície de ataque de um ambiente de software é a soma dos diferentes pontos (os "vetores de ataque") onde um usuário não autorizado (o "atacante") pode tentar inserir dados ou extrair dados de um ambiente. Manter a superfície de ataque o menor possível é uma medida básica de segurança.

Fonte: https://en.wikipedia.org/wiki/Attack_surface

Modelagem de ameaças

A modelagem de ameaças é um processo pelo qual ameaças potenciais podem ser identificadas, enumeradas e priorizadas - tudo do ponto de vista de um invasor hipotético.

O objetivo da modelagem de ameaças é fornecer aos defensores uma análise sistemática do perfil provável do invasor, dos vetores de ataque mais prováveis e dos ativos mais desejados por um invasor.

O modelo de ameaça baseado nesta apresentação é de um atacante com acesso local a máquina, servidores ou computadores pessoais rodando Linux.

Fonte: https://en.wikipedia.org/wiki/Threat_model

Modelagem de ameaças - Tipos de ameaças

Cada ameaça tem seus recursos e limitações. Lista de algumas possíveis ameaças:

1) Estado-nações
2) Criminosos
3) Funcionário interno malicioso (insider)
4) Hacktivistas
5) Malware / ransomwares
6) Acesso físico (USB e outros)
7) Usuário privilegiado e não privilegiado (núvem, containers e outros)
8) Acesso remoto / acesso local

Modelagem de ameaças - Caso Eddie Raymond Tipton

Caso conhecido de Eddie Raymond Tipton, ex-funcionário da Multi-State Lottery Association (MUSL), que modificou o gerador de número aleatórios (RNG) no jogo Hot Lotto, administrado por MUSL. Tipton foi condenado em outubro de 2015 por manipular um sorteio de 14,3 milhões de reais do jogo de loteria.

Lições:

A ameaça interna também precisa ser levada em consideração.

In 2017, Eddie Raymond Tipton, former information security director of the American Multi-State Lottery Association (MUSL), confessed to rigging a random number generator that he and two others used in multiple cases of fraud against state lotteries. He was sentenced to 25 years in prison.[1][2] Tipton was first convicted in October 2015 of rigging a $14.3 million drawing of MUSL's lottery game Hot Lotto. Eddie Tipton ultimately confessed to rigging lottery drawings in Iowa, Colorado, Wisconsin, Kansas and Oklahoma. Also involved in the scheme were his brother and former Texas Justice of the Peace Tommy Tipton, and Texas businessman Robert Rhodes.[1][2]

Fonte: https://en.wikipedia.org/wiki/Hot_Lotto_fraud_scandal

Modelagem de ameaças - Caso Eddie Raymond Tipton

A former Iowa Lottery employee who won millions by rigging the lottery system in several states was sentenced to 25 years in prison Tuesday.

Eddie Tipton, 54, who was a computer programmer in Multi State Lottery Association in Iowa, messed with the system to allow him and his brother Tommy to win jackpots in several states, according to the Des Moines *Register*.

Tipton was arrested for the fraud along with his brother in 2015. He pleaded guilty last June to installing software that enabled him to rig lottery numbers in his own favor starting in 2005. The hack affected systems in his home state, along with Colorado, Kansas, Oklahoma and Wisconsin. His brother Tommy also pleaded guilty to conspiracy to commit theft.

Fonte: http://time.com/4911802/eddie-tipton-powerball-lottery-prison-sentence/

Recomenda-se que tenha definido claramente quais são suas principais ameaças antes de qualquer investimento em segurança da informação.

PARTE 2

Reduzindo superfície de ataque

Redução da superfície de ataque

A redução de superfície de ataque tenta expor menos pontos de entrada ao *kernel* sem quebrar a funcionalidade legítima.

Pode incluir a remoção de código, a remoção do acesso a pontos de entrada ou a exposição seletiva de recursos.

Fonte: https://security.googleblog.com/2016/07/protecting-android-with-more-linux.html

Exemplos de vetores de ataques

Chamadas de sistema

As chamadas de sistema é uma das principais formas de se comunicar com o *kernel*.

Novas chamadas de sistema sendo adicionadas significa aumento da superfície de ataque. Mais código para um possível atacante interagir e descobrir problemas de segurança.

Módulos de terceiros

Protocolos de rede (ROSE, RDS, SCTP, IRDA), drivers de rede (wifi, ethernet), drivers de som, sistemas de arquivos.

Novas tecnologias e subsistemas

ABI x32, *Namespaces*, BPF, Netlink, XFRM, Crypto, Subsistema de rede, Subsistema de memória, Subsistema perf, Subsistema de chaveiro

E outras...

Chamadas de sistemas

Lista não exaustiva de algumas chamadas de sistema que podem ser abusadas maliciosamente.

userfaultfd() - Fornece primitiva interessante que pode ajudar durante a exploração de use-after-free;
kcmp() - Fornece primitivas interessantes para um possível atacante;
setns() - Subsistema de namespace permite um possível atacante obter privilégios elevados;
unshare() - Subsistema de namespace permite um possível atacante obter privilégios elevados;
ptrace() - Permite um usuário inspecionar e modificar outros processos; Vítima de várias vulnerabilidades publicamente conhecidas;
keyctl() - Subsistema de chaveiro vítima de vulnerabilidades publicamente conhecidas.
perf_event_open() - Subsistema perf, vítima de vulnerabilidades publicamente conhecidas.
modify_ldt() - Fornece primitiva interessante para um possível atacante.
bpf() - Fornece primitiva interessante para um possível atacante e já foi vítima de vulnerabilidades publicamente conhecidas;
kexec_load() - Fornece primitiva mais do que interessante para um possível atacante.

Caso não seja necessário ter essas chamadas de sistemas habilitadas em seu sistema, recomenda-se que sejam desabilitadas/bloqueadas. É possível realizar isso utilizando seccomp, que será discutido adiante, ou SELinux mas o ideal é que sejam desabilitadas diretamente no *kernel* recompilando-o.

Chamadas de sistemas

Chamadas de sistema disponíveis em um CentOS 7 kernel versão 3.10.

```
301     common  fanotify_mark           sys_fanotify_mark
302     common  prlimit64               sys_prlimit64
303     common  name_to_handle_at       sys_name_to_handle_at
304     common  open_by_handle_at       sys_open_by_handle_at
305     common  clock_adjtime           sys_clock_adjtime
306     common  syncfs                  sys_syncfs
307     64      sendmmsg                sys_sendmmsg
308     common  setns                   sys_setns
309     common  getcpu                  sys_getcpu
310     64      process_vm_readv        sys_process_vm_readv
311     64      process_vm_writev       sys_process_vm_writev
312     common  kcmp                    sys_kcmp
313     common  finit_module            sys_finit_module
316     common  renameat2               sys_renameat2
319     common  memfd_create            sys_memfd_create
320     common  kexec_file_load         sys_kexec_file_load
323     common  userfaultfd             sys_userfaultfd
```

Chamadas de sistemas

Chamadas de sistema disponíveis em *upstream*, linux *kernel* versão 5.0. 15 chamadas de sistema adicionais comparado com o *kernel* 3.10

```
334   323     common  userfaultfd             __x64_sys_userfaultfd
335   324     common  membarrier              __x64_sys_membarrier
336   325     common  mlock2                  __x64_sys_mlock2
337   326     common  copy_file_range         __x64_sys_copy_file_range
338   327     64      preadv2                 __x64_sys_preadv2
339   328     64      pwritev2                __x64_sys_pwritev2
340   329     common  pkey_mprotect           __x64_sys_pkey_mprotect
341   330     common  pkey_alloc              __x64_sys_pkey_alloc
342   331     common  pkey_free               __x64_sys_pkey_free
343   332     common  statx                   __x64_sys_statx
344   333     common  io_pgetevents           __x64_sys_io_pgetevents
345   334     common  rseq                    __x64_sys_rseq
346   # don't use numbers 387 through 423, add new calls after the last
347   # 'common' entry
348   424     common  pidfd_send_signal       __x64_sys_pidfd_send_signal
349   425     common  io_uring_setup          __x64_sys_io_uring_setup
350   426     common  io_uring_enter          __x64_sys_io_uring_enter
351   427     common  io_uring_register       __x64_sys_io_uring_register
352
```

Fonte: https://github.com/torvalds/linux/blob/master/arch/x86/entry/syscalls/syscall_64.tbl

SECCOMP

A chamada de sistema seccomp() pode ser utilizada para restringir o acesso a outras chamadas de sistema. Ela suporta dois modos de operação:

SECCOMP_SET_MODE_STRICT

Após habilitado, permite somente read(), write(), _exit() e sigreturn().

SECCOMP_SET_MODE_FILTER

Neste modo as chamadas de sistema permitidas são definidas por um filtro BPF passado via argumento. Permite ter mais controle sobre quais chamadas de sistemas são permitidas.

Recomenda-se o uso de seccomp caso a aplicação suportar. Docker, Chromium, Systemd, OpenSSH, Firejail e outras aplicações tem suporte a seccomp.

Berkeley Packet Filter (BPF)

O BPF suporta pacotes de filtragem, permitindo que um processo do espaço de usuário forneça um programa de filtro que especifica quais pacotes ele deseja receber. BPF hoje pode ser usado para múltiplos propósitos, inclusive filtragem de chamadas de sistema.

Existe também o BPF JIT, tecnologia que traduz/compila filtro BPF para um correspondente em linguagem de máquina visando aumento de performance. BPF JIT fornece primitivas/condições interessantes para um possível atacante.

Usado por:

bpf()

setsockopt() - SO_ATTACH_FILTER

seccomp-bpf

Networking (XDP, tc, tcpdump e outras)

Berkeley Packet Filter (BPF) - Algumas vulnerabilidades

ID ▼	Status ▼	Restrict ▼	Reported ▼	Vendor ▼	Product ▼	Finder ▼	Summary + Labels ▼
808	Fixed	----	2016-Apr-26	Linux	Linux	jannh	Linux: UAF via double-fdput() in bpf(BPF_PROG_LOAD) error path CCProjectZeroMembers
809	Fixed	----	2016-Apr-27	Linux	Linux	jannh	Linux: reference count overflow using BPF maps CCProjectZeroMembers
1251	Fixed	----	2017-May-6	Linux	Linux	jannh	Linux: eBPF verifier log leaks lower half of map pointer CCProjectZeroMembers
1454	Fixed	----	2017-Dec-4	Linux	Linux	jannh	arbitrary read+write via incorrect range tracking in eBPF CCProjectZeroMembers
1496	Fixed	----	2017-Dec-6	Linux	Linux	jannh	eBPF verifier bug backported to 4.9-stable CCProjectZeroMembers
1528	Fixed	----	2018-Feb-6	AMD, Intel	----	jannh	speculative execution, variant 4: speculative store bypass CCProjectZeroMembers
1655	Fixed	----	2018-Sep-5	Linux	Linux	jannh	Linux: kernel ptr leak via BPF: broken subtraction check CCProjectZeroMembers
1657	Fixed	----	2018-Sep-5	Linux	Linux	jannh	Linux: semi-arbitrary task stack read on ARM64 (and x86) via /proc/$pid/stack CCProjectZeroMembers
1686	Fixed	----	2018-Oct-3	Linux	Linux	jannh	Linux: bpf verifier: 32-bit RSH verification doesn't truncate input before the ALU op CCProjectZeroMembers
1711	Fixed	----	2018-Nov-5	Linux	Linux	jannh	Linux: eBPF Spectre v1 mitigation is insufficient CCProjectZeroMembers

Berkeley Packet Filter (BPF)

Recomenda-se se possível desabilitar a chamada de sistema bpf() totalmente ou então desabilitar para usuários não privilegiados.

kernel.unprivileged_bpf_disabled

Restringe a chamada de sistema bpf() somente para usuários privilegiados.

Módulos de terceiros

Em computação, um módulo carregável do núcleo é um arquivo objeto que contém código para estender *(aumentar a superfície de ataque)* o núcleo em execução, ou o chamado núcleo base, de um sistema operacional.

Exemplos:

Protocolos de rede

BLUETOOTH (AF_BLUETOOTH), INFRARED (AF_IRDA), IPV6 (AF_INET6), ECONET (AF_ECONET), X25 (AF_X25), AX25 (AF_AX25), ROSE (AF_ROSE), RDS (AF_RDS), IGMP (AF_IGMP), DCCP (AF_DCCP), SCTP (AF_SCTP), LLC (AF_LLC), UNIX (AF_UNIX), PACKET (AF_PACKET)

Drivers de rede com fio (wired), rede sem fio (wireless), vídeo e outros.

Sistemas de arquivos

REISERFS, OVERLAYFS, BTRFS, SQUASHFS, NFS, CIFS, SMBFS, EXT3, EXT4, XFS, NTFS, ECRYPTFS

Carregamento automático de módulos - *module autoloading*

Mesmo se módulos considerados frágeis não estiverem em uso no sistema, um possível atacante pode abusar do recurso de carregamento automático de módulos.

Recomenda-se se possível desabilitar o carregamento de módulos após o boot ou no mínimo colocar em lista negra módulos considerados de segurança frágil.

Exemplo de uso:

Desde Ubuntu 11.04, ax25, netrom, x25, rose, decnet, econet, rds e af_802154 estão proibidos de serem carregados automaticamente.

Blacklist Rare Protocols

Normally the kernel allows all network protocols to be autoloaded on demand via the `MODULE_ALIAS_NETPROTO(PF_...)` macros. Since many of these protocols are old, rare, or generally of little use to the average Ubuntu user and may contain undiscovered exploitable vulnerabilities, they have been blacklisted since Ubuntu 11.04. These include: ax25, netrom, x25, rose, decnet, econet, rds, and af_802154. If any of the protocols are needed, they can speficially loaded via modprobe, or the `/etc/modprobe.d/blacklist-rare-network.conf` file can be updated to remove the blacklist entry.

See test-kernel-security.py for regression tests.

Módulos de terceiros

Desabilitar carregamento de módulos do *kernel* mesmo para usuário root.

Sysctl

```
sysctl kernel.modules_disabled=1
```

Sistema de arquivo:

```
echo 1 > /proc/sys/kernel/modules_disabled
```

CVE-2017-6074 - Vulnerabilidade no protocolo DCCP - Android

EDITION: US

ZDNet

VIDEOS SMART CITY WINDOWS 10 CLOUD INNOVATION SECURITY ENTERPRISE IOT MORE NEWSLETTER

MUST READ WHAT TO EXPECT FROM THE WINDOWS 10 CREATORS UPDATE

Linux's decade-old flaw: Major distros move to patch serious kernel bug

Google fuzzer helps find 11-year-old memory-corruption flaw in the Linux kernel.

By Liam Tung | February 23, 2017 -- 14:50 GMT (06:50 PST) | Topic: Security

CVE-2017-6074: DCCP double-

Layers of attack surface reduction

- Not compiled into Android common kernels
- Even if compiled in, not reachable due to SELinux restrictions.
- “dodged a bullet” -> “working as intended”

Networking Protocols

- Only a whitelist of socket families are allowed
  - Netlink Route Sockets
  - Ping Sockets
  - TCP / UDP Sockets
  - Unix stream and datagram sockets
- Whitelist allowed ioctls

```
# Restrict socket ioctls. Either
# 1. disallow privileged ioctls,
# 2. disallow the ioctl permission, or
# 3. disallow the socket class.

neverallowxperm untrusted_app domain:{ rawip_socket
tcp_socket udp_socket } ioctl priv_sock_ioctls;

neverallow untrusted_app *:{ netlink_route_socket
netlink_selinux_socket } ioctl;

neverallow untrusted_app *:{
  socket netlink_socket packet_socket key_socket
  appletalk_socket netlink_firewall_socket
  netlink_tcpdiag_socket netlink_nflog_socket
  netlink_xfrm_socket netlink_audit_socket
  netlink_ip6fw_socket
  netlink_dnrt_socket netlink_kobject_uevent_socket
  tun_socket netlink_iscsi_socket
  netlink_fib_lookup_socket netlink_connector_socket
  netlink_netfilter_socket netlink_generic_socket
  netlink_scsitransport_socket
  netlink_rdma_socket netlink_crypto_socket
} *;
```

Fonte: https://www.blackhat.com/docs/us-17/thursday/us-17-Kralevich-Honey-I-Shrunk-The-Attack-Surface-Adventures-In-Android-Security-Hardening.pdf

Namespaces

Namespaces é um recurso do kernel do Linux que particiona os recursos do *kernel,* de modo que um conjunto de processos vê um conjunto de recursos, enquanto outro conjunto de processos vê um conjunto diferente de recursos.

user namespace
mount namespace
net namespace
PID namespace
IPC namespace
UTS namespace

Várias vulnerabilidades afetaram o subsistema namespace mas o namespace de usuário (*user namespace*) merece um olhar mais detalhado. Este *namespace* permite um usuário não privilegiado obter privilégio root dentro de um contêiner. Este usuário privilegiado pode ter uma maior superfície de ataque.

Chamadas de sistemas que permitem criar um novo namespace:

unshare()
setns()
clone()

Fonte: https://en.wikipedia.org/wiki/Linux_namespaces

Namespaces - *Exploits* usando *user namespace* CLONE_NEWUSER

https://www.exploit-db.com/exploits/44049 - XFRM Ubuntu 17.04

https://www.exploit-db.com/exploits/43418 - UFO to NON-UFO 4.4.0-83 / < 4.8.0-58 (Ubuntu 14.04/16.04)

https://www.exploit-db.com/exploits/41994 - AF_PACKET 4.8.0-41-generic (Ubuntu)

https://www.exploit-db.com/exploits/43127 - waitid() 4.13 (Ubuntu 17.10)

https://www.exploit-db.com/exploits/41762 - User Namespace Overlayfs (Ubuntu 14.04/15.10)

https://www.exploit-db.com/exploits/44300 - Netfilter target_offset 4.4.0-21 (Ubuntu 16.04 x64)

https://www.exploit-db.com/exploits/38390 - CLONE_NEWUSER|CLONE_FS 3.0 < 3.3.5

https://www.exploit-db.com/exploits/41458 - Linux Kernel 4.4.0 (Ubuntu) DCCP Double-Free Privilege Escalation

Namespaces

Desabilitar *user namespace* completamente ou permitir somente para usuários privilegiados

kernel.unprivileged_userns_clone

Não disponível em *upstream*, disponível via *patch* por mantenedores Debian
Permite ou não permite

/proc/sys/kernel/userns_restrict

Proposta de *patch*
Três opções:

0 - *user namespace* habilitado para todos usuários
1 - *user namespace* somente usuários privilegiados
2 - *user namespace* desabilitado completamente

Capabilities

Linux *capabilities* fornecem um controle refinado sobre permissões de superusuário (root), permitindo que o uso do usuário root seja evitado.

Recomendado para evitar o uso do usuário root desnecessariamente mas seu uso incorreto pode aumentar a superfície de ataque ao permitir que usuários não privilegiados alcancem/acessem código que deveria estar exposto somente para usuário administrativo (root).

Recomenda-se validar quais as *capabilities* atribuídas pelo sistema.

Exemplo de uso de *capability*:

ping - CAP_NET_RAW

Capabilities

Algumas *capabilities*:

CAP_SYS_ADMIN

O novo root;

CAP_NET_ADMIN

Realiza várias operações de rede como: Configurar interface, habilitar modo promíscuo, modificar tabela de roteamento;

CAP_NET_RAW

Usar *sockets* RAW (AF_RAW) e PACKET (AF_PACKET);

CAP_SYS_MODULE

Carregar e descarregar módulos do *kernel*;

Capabilities - Exemplo da vulnerabilidade CVE-2017-7184 no Android

Vulnerabilidade utilizada para comprometer Ubuntu em competição hacker Pwn2own em 2017.

Efetivamente inacessível no Android.

CVE-2017-7184: xfrm kernel heap out-of-bounds access

- Compiled into Android kernels
- Requires CAP_NET_ADMIN
  - Available to lots of processes on Android.
- Requires netlink_xfrm_socket
  - Who has it?

CVE-2017-7184: xfrm kernel heap

- Reachability:
  - Only available to one process!
  - Effectively unreachable.

Fonte: https://www.blackhat.com/docs/us-17/thursday/us-17-Kralevich-Honey-I-Shrunk-The-Attack-Surface-Adventures-In-Android-Security-Hardening.pdf

Capabilities, bpf e seccomp - Vulnerabilidade CVE-2017-16995

What Does This Mean for Containers?

As for container security, we are in a relatively good place. eBPF programs are loaded by means of the *bpf()* system call.

The access to bpf() system call is gated by CAP_SYS_ADMIN capability, which is not assigned by default to containers.
On the other hand - if the OS is configured to allow unprivileged users to run bpf programs (which is actually the default on most Linux distributions), then there is no need for the CAP_SYS_ADMIN capability since the bpf() system call is available within containers by default!

Docker applies seccomp as second layer of defense: when enabled, there is a default seccomp profile that blocks the bpf syscall. One caveat is when you run with CAP_SYS_ADMIN -- in that case the seccomp profile does not block the bpf() system call.

Fonte: https://blog.aquasec.com/ebpf-vulnerability-cve-2017-16995-when-the-doorman-becomes-the-backdoor

MUITO OBRIGADO!

Dúvidas?

SECURITY INTELLIGENCE