

HORSE PILL

A New Kind of
Linux Rootkit





\$ whoami

Michael Leibowitz

(@r00tkillah)

- Intel Red Team
- NSA Playset
- stuff

Overview

- What is a rootkit
- History of rootkits
- How your computer boots
- What is/isn't protected
- Containers
- Putting it together
- Demo
- Properties
- Detection
- Mitigation

What is a rootkit?

- Post Exploitation
- Persistent Access
- Covert Access



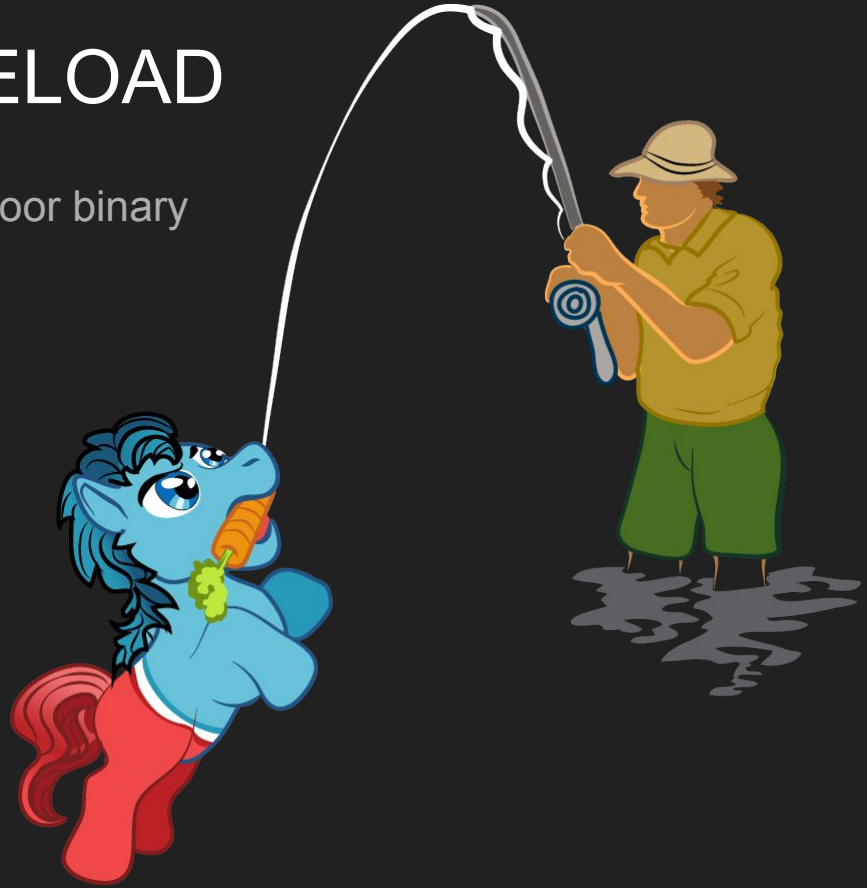
Historical Rootkits - backdoored commands

1. Backdoor `inetd`
2. Blind all tools to see rootkit
 - `ps`
 - `sum`
 - `top`
 - `find`
 - `lsf`
 - `netstat`
3. Connect to shell served by `inetd`
4. ???
5. **PROFIT!**




Historical Rootkits - LD_PRELOAD

1. Add malicious library to `ld.so.preload`, backdoor binary
2. Hook
 - a. `stat()`
 - b. `open()`
 - c. `opendir()`
 - d. `readdir()`
 - e. `unlink()`
3. Enjoy your shell
4. ???
5. Profit!!!




Historical Rootkits - Kernel Module

1. Insert malicious kernel module
2. Make invisible
 - a. Network connections
 - b. Files
 - c. Processes
 - d. Module itself
 - e.  *Desirable Other Evil*
3. Enjoy your shell
4. ???
5. Profit!!!



Historical Rootkits - /dev/mem

1. Open memory and shove in malicious code
2. Make invisible
 - a. Network connections
 - b. Files
 - c. Processes
 -  d. *Desirable Other Evil*
3. Enjoy your shell
4. ???
5. Profit!!!



What is a container?

Namespaces and cgroups

Hierarchies and non-hierarchies



Clone, man. Man clone(2).

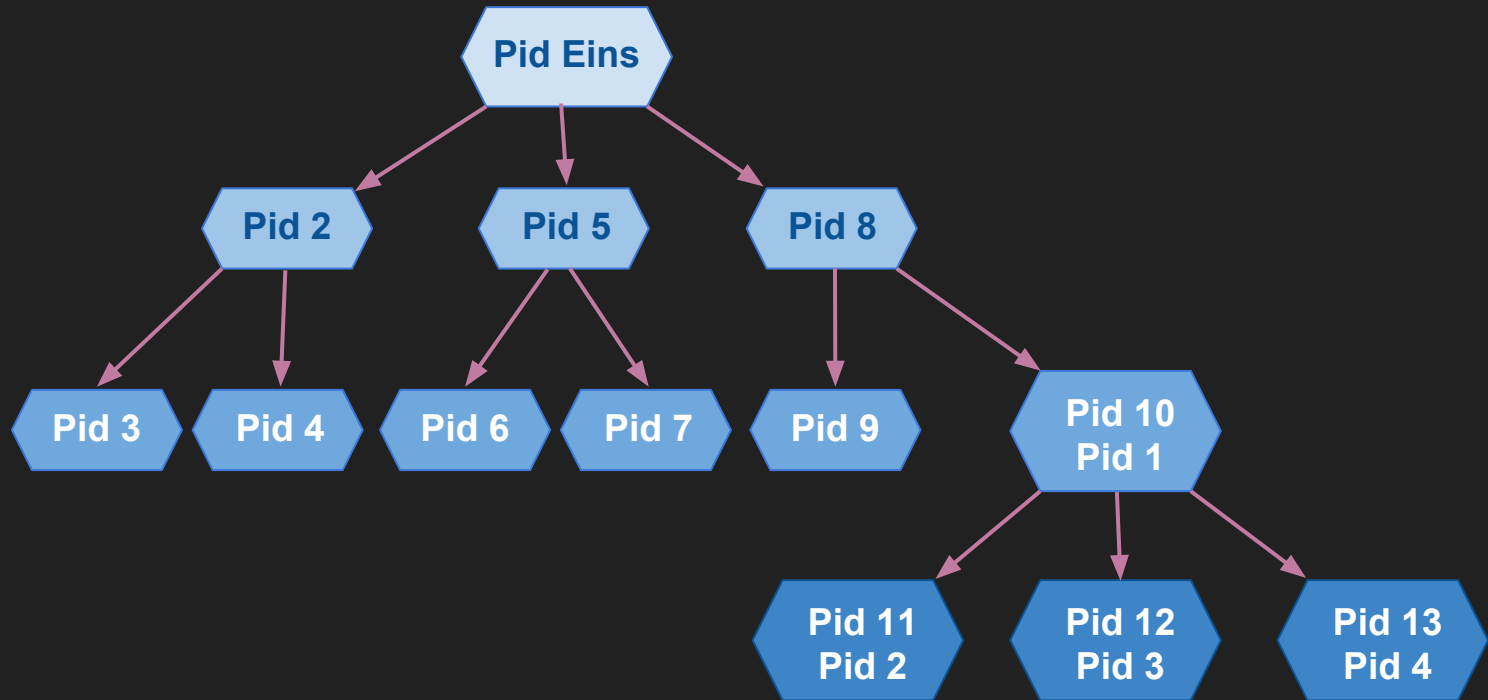
- Namespace creation controlled with unshare(2) and clone(2)
- namespaces traversed with setns(2)

```
root@gtfo:~# ls -l /proc/1/ns
total 0
lrwxrwxrwx 1 root root 0 Jul  8 16:47 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 root root 0 Jul  8 16:47 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 root root 0 Jul  8 16:47 net -> net:[4026531969]
lrwxrwxrwx 1 root root 0 Jul  8 16:47 pid -> pid:[4026531836]
lrwxrwxrwx 1 root root 0 Jul  8 16:47 user -> user:[4026531837]
lrwxrwxrwx 1 root root 0 Jul  8 16:47 uts -> uts:[4026531838]
```

Namespace Magic Numbers

```
root@gtfo:/usr/src/linux# cat -n include/linux/proc_ns.h | grep -A2 -B8 PROC_PID_INIT_INO
31     /*
32     * We always define these enumerators
33     */
34     enum {
35         PROC_ROOT_INO          = 1,
36         PROC_IPC_INIT_INO     = 0xEFFFFFFFU,
37         PROC_UTS_INIT_INO     = 0xEFFFFFFEU,
38         PROC_USER_INIT_INO    = 0xEFFFFFFDU,
39         PROC_PID_INIT_INO     = 0xEFFFFFFCU,
40         PROC_CGROUP_INIT_INO = 0xEFFFFFFBU,
41     };
```

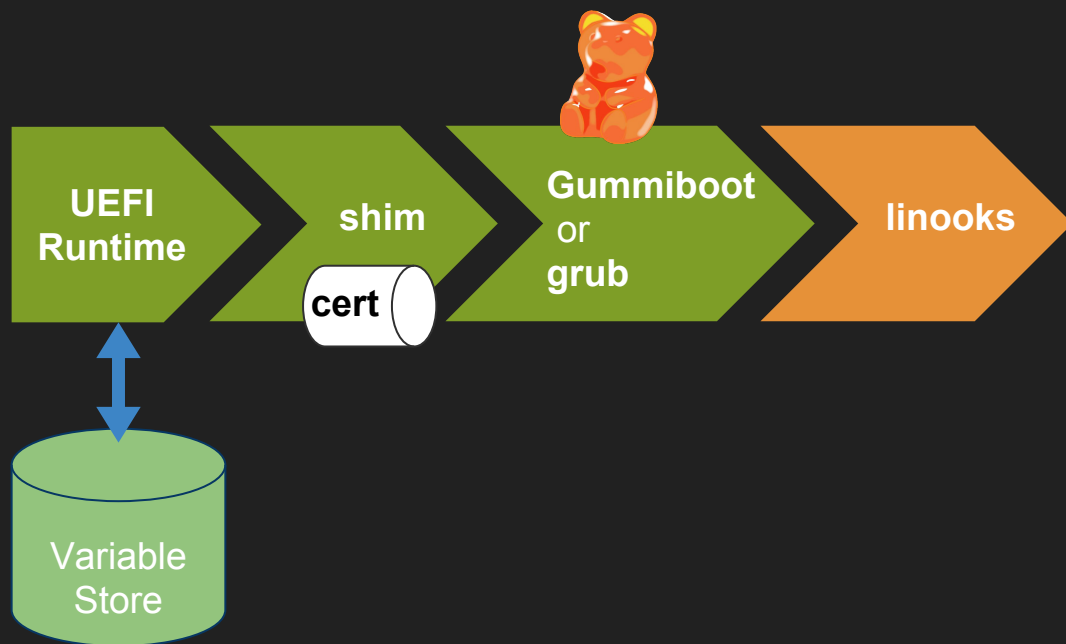
Process Hierarchies



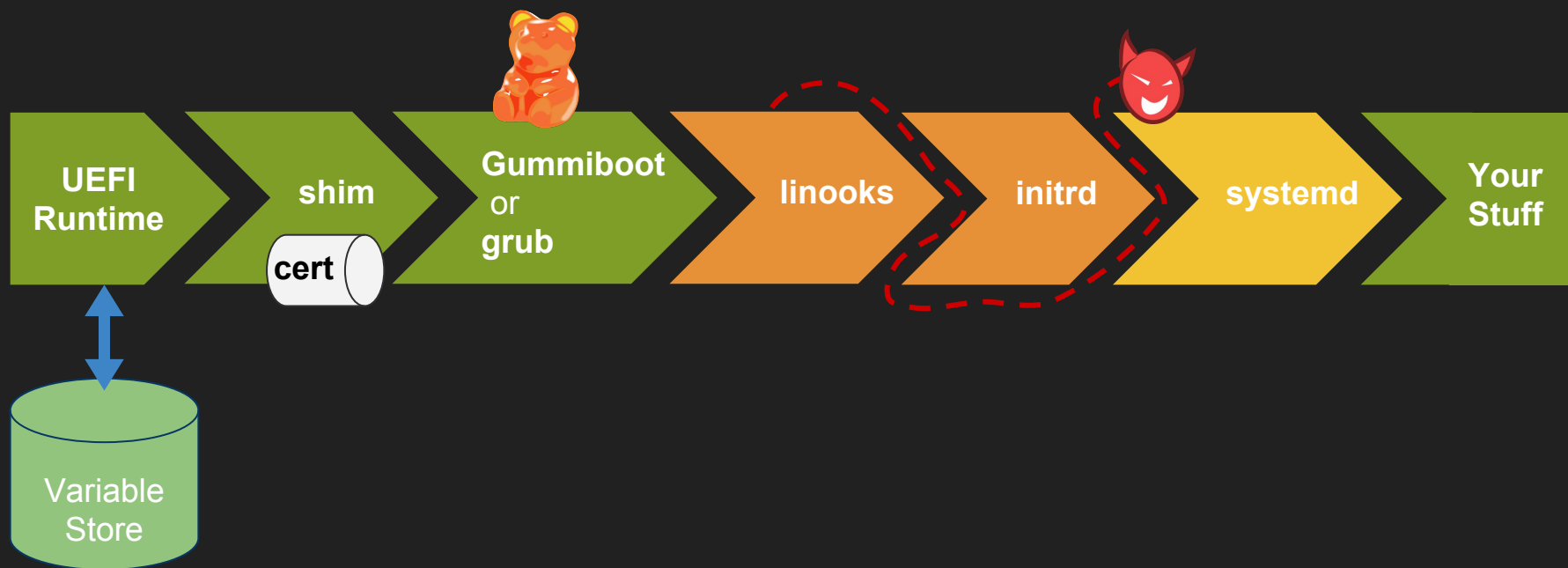
How Your Computer Boots

1. UEFI
2. Shim
3. Gummiboot
4. Kernel
5. initrd
6. systemd

How Your Computer Boots



How Your Computer Boots



Protected / **Not** protected

Bootloader

Kernel

Modules

Initrd

Rootfs



Anatomy of a Ramdisk



What Your Ramdisk is Supposed to do

1. Load necessary modules/respond to hotplug events
2. Cryptsetup *<optional>*
3. Find and mount rootfs
4. Clean up initrd
5. Exec init
6. ???
7. Profit!!!

Anatomy of a Ramdisk (now)



What Your Ramdisk Does **Now**

1. Load modules/hotplug events
2. Cryptsetup
3. Find and mount rootfs
4. Enumerate kernel threads


What Your Ramdisk Does Now

1. Load modules/hotplug events
 2. Cryptsetup
 3. Find and mount rootfs
 4. Enumerate kernel threads
 5. Clone (`CLONE_NEWPID, CLONE_NEWNS`)
- 
1. Remount proc
 2. Make fake kernel threads
 3. Clean up initrd
 4. Exec init

What Your Ramdisk Does Now

1. Load modules/hotplug events
 2. Cryptsetup
 3. Find and mount rootfs
 4. Enumerate kernel threads
 5. Clone (`CLONE_NEWPID, CLONE_NEWNS`)
 6. Remount root
 7. Mount scratch space
- 
1. Remount proc
 2. Make fake kernel threads
 3. Clean up initrd
 4. Exec init

What Your Ramdisk Does Now

1. Load modules/hotplug events
 2. Cryptsetup
 3. Find and mount rootfs
 4. Enumerate kernel threads
 5. Clone (`CLONE_NEWPID, CLONE_NEWNS`)
 6. Remount root
 7. Mount scratch space
 8. `fork()`
 - a. Hook `initrd` updates
 - b. Backdoor shell
 9. `waitpid()`
 10. shutdown/reboot
- 
1. Remount `proc`
 2. Make fake kernel threads
 3. Clean up `initrd`
 4. Exec `init`

Kernel Threads

```
root@gtfo:~# ps auxf | head -n 20
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	2	0.0	0.0	0	0	?	S	Jul09	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	Jul09	0:00	_ [ksoftirqd/0]
root	5	0.0	0.0	0	0	?	S<	Jul09	0:00	_ [kworker/0:0H]
root	7	0.0	0.0	0	0	?	S	Jul09	0:29	_ [rcu_sched]
root	8	0.0	0.0	0	0	?	S	Jul09	0:00	_ [rcu_bh]
root	9	0.0	0.0	0	0	?	S	Jul09	0:16	_ [rcuos/0]
root	10	0.0	0.0	0	0	?	S	Jul09	0:00	_ [rcuob/0]
root	11	0.0	0.0	0	0	?	S	Jul09	0:00	_ [migration/0]
root	12	0.0	0.0	0	0	?	S	Jul09	0:00	_ [watchdog/0]
root	13	0.0	0.0	0	0	?	S	Jul09	0:00	_ [watchdog/1]
root	14	0.0	0.0	0	0	?	S	Jul09	0:00	_ [migration/1]
root	15	0.0	0.0	0	0	?	S	Jul09	0:00	_ [ksoftirqd/1]

prctl/setting process name

```
prctl_map = (struct prctl_mm_map) {  
    ...  
    .arg_start = arg_start,  
    .arg_end = arg_end,  
    ...  
};
```

```
ret = prctl(PR_SET_MM, PR_SET_MM_MAP, (long) &prctl_map, sizeof(prctl_map), 0);
```

```
prctl(PR_SET_NAME, (unsigned long)buf, 0, 0, 0) < 0);
```

Putting it Together: Coverttness

Goal

- A. Processes Invisibility
- B. Storage Invisibility
- C. Networking Invisibility



Hiding Network Traffic

```
root@gtfo:~# head -n1 /proc/net/tcp ; cat /proc/net/tcp | grep 0101007F:0035
```

```
s1 local_address      rem_address  ...  inode  ...  
3: 0101007F:0035      00000000:0000 ...  20041  ...
```

```
root@gtfo:~# ls -l /proc/1894/fd | grep 20041  
lrwx----- 1 root root 64 Jul 17 10:23 5 -> socket:[20041]
```

Putting it together: Persistence

How do we get our malicious binary into ramdisks on upgrade?

1. Assemble `initrd` contents into `tmpdir`
2. Splat 🐎🔑 over `run-init`
3. Archive and compress `tmpdir`
4. ???
5. Profit!!!!

Demo



Properties

Covert

- Processes
- Networking
- Storage

Persistent



Detection

- /proc/<pid>/ns links
- Kernel threads proc entries (ppid != 0)
- Audit
- External examination



What we can do to fix this

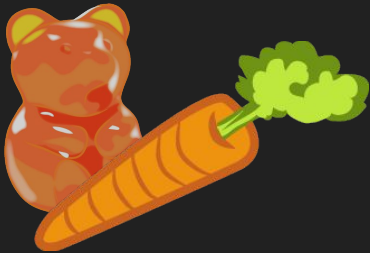
STOP

assembling
ramdisks on
systems!



Conclusion

- What is a rootkit
- History of rootkits
- How your computer boots
- What is/isn't protected
- Containers
- Putting it together
- Demo
- Properties
- Detection
- Mitigation





Questions?

Michael Leibowitz

(@r00tkillah)