



# RUMP

## CONTAINER ESCAPE





# THE STORY STARTS WITH A JENKINS...

- Internal Pentest
- Reachable Jenkins
- Remote command execution using groovy script console





# BUT UNLUCKY..

- The jenkins server runs in a docker container:

```
jenkins@7d1b89562981:/$ cat /proc/1/cgroup |grep docker
11:memory:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
10:cpuset:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
9:net_cls,net_prio:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
8:freezer:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
7:blkio:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
6:hugetlb:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
5:devices:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
4:perf_event:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
3:pids:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
2:cpu,cpuacct:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
1:name=systemd:/docker/7d1b89562981ce0ac2103a673cd473ad87695bfdab215cc117561d47b438d77e
```

- The docker container is using SECCOMP

```
jenkins@7d1b89562981:/$ cat /proc/1/status|grep Seccomp
Seccomp:          2
```

- However:

Kernel is not up-to-date: “Kernel Ubuntu 4.4.0.79 (june 2017)”





2017: January February March April May June July August September October November December CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : [CVE Number Descending](#) [CVE Number Ascending](#) [CVSS Score Descending](#) [Number Of Exploits Descending](#)

[Copy Results](#) [Download Results](#)

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.
1	<a href="#">CVE-2017-1000255</a>	<a href="#">787</a>			2017-10-30	2018-04-10	6.6	None	Local	Low	Not required	None	Complete
<p>On Linux running on PowerPC hardware (Power8 or later) a user process can craft a signal frame and then do a sigreturn so that the kernel will take an exception (interrupt), and use the r1 value *from the signal frame* as stack pointer. As part of the exception entry the content of the signal frame is written to the kernel stack, allowing an attacker to overwrite arbitrary locations with arbitrary values. The exception handling does produce an oops, but only after kernel memory has been over written. This flaw was introduced in commit: "5d176f751ee3 (powerpc: tm: Enable transactional memory (TM) lazily for userspace)" which was merged into v4.9-rc1. Please note that kernels built with CONFIG_PPC_TRANSACTIONAL_MEM=n are not vulnerable.</p>													
2	<a href="#">CVE-2017-1000112</a>	<a href="#">362</a>		Mem. Corr.	2017-10-04	2017-12-22	6.9	None	Local	Medium	Not required	Complete	Complete
<p>Linux kernel: Exploitable memory corruption due to UFO to non-UFO path switch. When building a UFO packet with MSG_MORE __ip_append_data() calls ip_ufo_append_data() to append. However in between two send() calls the append path can be switched from UFO to non-UFO one, which leads to a memory corruption. In case UFO packet lengths exceeds MTU, copy = maxfraglen - skb-&gt;len becomes negative on the non-UFO path and the brainiac allocate new skb is taken. This triggers fragmentation and computation of fraggap = skb_prev-&gt;len - maxfraglen. Fraggap can exceed MTU, causing copy = datalen - transhdrlen - fraggap to become negative. Subsequently skb_copy_and_csum_bits() writes out-of-bounds. A similar issue is present in IPv6 code. The bug was introduced in e89e9cf539a2 ("[IPv4/IPv6]: UFO Scatter-gather approach") on Oct 18 2005.</p>													
3	<a href="#">CVE-2017-1000111</a>	<a href="#">264</a>			2017-10-04	2017-12-06	7.2	None	Local	Low	Not required	Complete	Complete
<p>Linux kernel: heap out-of-bounds in AF_PACKET sockets. This new issue is analogous to previously disclosed CVE-2016-8655. In both cases, a socket option that changes socket state may race with safety checks in packet_setsockopt(). Previously with PACKET_VERSION. This time with PACKET_RESERVE. The solution is similar: lock the socket for the update. This issue may be exploitable, we did not investigate further. As this issue affects PF_PACKET it requires CAP_NET_RAW in the process namespace. But note that with user namespaces enabled, any process can create a namespace in which it has CAP_NET_RAW.</p>													
4	<a href="#">CVE-2017-15951</a>	<a href="#">20</a>		DoS	2017-10-27	2017-11-13	7.2	None	Local	Low	Not required	Complete	Complete
<p>The KEYS subsystem in the Linux kernel before 4.13.10 does not correctly synchronize the actions of updating versus finding a key in the "negative" state to avoid a race condition, which allows local users to cause a denial of service or possibly have unspecified other impact via crafted system calls.</p>													
5	<a href="#">CVE-2017-15649</a>	<a href="#">362</a>		+Priv	2017-10-19	2018-02-03	4.6	None	Local	Low	Not required	Partial	Partial
<p>net/packet/af_packet.c in the Linux kernel before 4.13.6 allows local users to gain privileges via crafted system calls that trigger mishandling of packet_fanout data structures, because of a race condition (involving fanout_a and packet_do_bind) that leads to a use-after-free, a different vulnerability than CVE-2017-6346.</p>													



# THE VULNERABILITY

- Vulnerability reported by Andrey Konovalov (Xairy) from projet zero

*From: Andrey Konovalov <andreyknvl () gmail com>*

*Date: Thu, 10 Aug 2017 22:55:29 +0200*

Hi!

syzkaller found an exploitable memory corruption in UFO code in the Linux kernel, the details are below.

### Bug details

When building a UFO packet with MSG\_MORE \_\_ip\_append\_data() calls ip\_ufo\_append\_data() to append. However in between two send() calls, the append path can be switched from UFO to non-UFO one, which leads to a memory corruption.

In case UFO packet lengths exceeds MTU, copy = maxfraglen - skb->len becomes negative on the non-UFO path and the branch to allocate new skb is taken. This triggers fragmentation and computation of fraggap = skb\_prev->len - maxfraglen. Fraggap can exceed MTU, causing copy = datalen - transhdrlen - fraggap to become negative. Subsequently skb\_copy\_and\_csum\_bits() writes out-of-bounds.

A similar issue is present in IPv6 code.

The bug was introduced in e89e9cf539a2 ("[IPv4/IPv6]: UFO Scatter-gather approach") on Oct 18 2005.

The fix has been submitted to netdev [1] and should be committed to mainline and to stable kernels soon. David has also sent an RFC series to remove UFO completely [2], which should be merged in 4.14.

- POC available at <https://github.com/xairy/kernel-exploits/blob/master/CVE-2017-1000112/poc.c>




# WTF IS UFO ?

- **UFO stands for UDP fragmentation offload:**

## UDP Fragmentation Offload

=====

UDP fragmentation offload allows a device to fragment an oversized UDP datagram into multiple IPv4 fragments. Many of the requirements for UDP fragmentation offload are the same as TSO. However the IPv4 ID for fragments should not increment as a single IPv4 datagram is fragmented.

IPv4/IPv6: UFO (UDP Fragmentation Offload) Scatter-gather approach: UFO is a feature wherein the Linux kernel network stack will offload the IP fragmentation functionality of large UDP datagram to hardware. This will reduce the overhead of stack in fragmenting the large UDP datagram to MTU sized packets  [commit](#)

## TL;DR

- **send(s, buffer, sizer , MSG\_MORE);**
- **Send(s, buffer, 1, 0);**



# UNPRIVILEGED USER NAMESPACE?

*From: Andrey Konovalov <andreyknvl () gmail com>*

*Date: Thu, 10 Aug 2017 22:55:29 +0200*

Hi!

syzkaller found an exploitable memory corruption in UFO code in the Linux kernel, the details are below.

### Bug details

When building a UFO packet with MSG\_MORE \_\_ip\_append\_data() calls ip\_ufo\_append\_data() to append. However in between two send() calls, the append path can be switched from UFO to non-UFO one, which leads to a memory corruption.

In case UFO packet lengths exceeds MTU, copy = maxfraglen - skb->len becomes negative on the non-UFO path and the branch to allocate new skb is taken. This triggers fragmentation and computation of fraggap = skb\_prev->len - maxfraglen. Fraggap can exceed MTU, causing copy = datalen - transhdrlen - fraggap to become negative. Subsequently skb\_copy\_and\_csum\_bits() writes out-of-bounds.

A similar issue is present in IPv6 code.

The bug was introduced in e89e9cf539a2 ("[IPv4/IPv6]: UFO Scatter-gather approach") on Oct 18 2005.

The fix has been submitted to netdev [1] and should be committed to mainline and to stable kernels soon. David has also sent an RFC series to remove UFO completely [2], which should be merged in 4.14.

If unprivileged user namespaces are available, this bug can be exploited to gain root privileges. I'll share the details and the exploit in a few days.

Thanks!



# UNPRIVILEGED USER NAMESPACE?

CVE-2017-1000112

English ▼

An exploitable memory corruption flaw was found in the Linux kernel. The append path can be erroneously switched from UFO to non-UFO in `ip_ufo_append_data()` when building an UFO packet with `MSG_MORE` option. **If unprivileged user namespaces are available, this flaw can be exploited to gain root privileges.**

- **Syscall unshare is prevented by seccomp default profile**

**To sum-up the exploitation condition are the following:**

- **Network interface with a MTU  $\leq 1500$**
- **Network interface with UFO enable ( by default on loopback)**

**Username space are used by the exploit to reduce the MTU of the loopback interface... can we still exploit ?**





# GOOD NEWS EVERYONE !

- LXC and Docker container create an interface with UFO enable

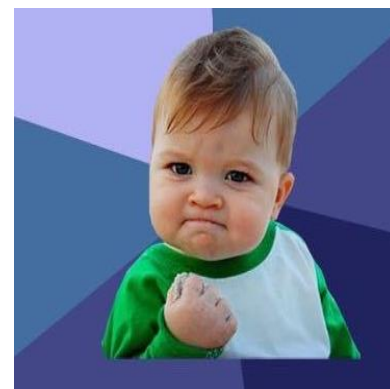
```
root@7d1b89562981:/# ethtool -k eth0
Features for eth0:
rx-checksumming: on
tx-checksumming: on
    tx-checksum-ipv4: off [fixed]
    tx-checksum-ip-generic: on
    tx-checksum-ipv6: off [fixed]
    tx-checksum-fcoe-crc: off [fixed]
    tx-checksum-sctp: off [fixed]
scatter-gather: on
    tx-scatter-gather: on
    tx-scatter-gather-fraglist: on
tcp-segmentation-offload: on
    tx-tcp-segmentation: on
    tx-tcp-ecn-segmentation: on
    tx-tcp6-segmentation: on
udp-fragmentation-offload: on
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off [fixed]
```





# TO SUMMARIZE THE RECIPE

- Look for an interface with UFO enable.
- Create a socket with UFO that will be routed by the kernel to this interface.
- Add the SMEP gadgets matching our kernel version.
- Change the payload to add a call to `call_usermodehelper (run_cmd)`
- Profit.



All the credit goes to Andrey Kononov



# KERNEL VERSION & EPILOGUE

- Epilogue: Nothing interesting in the docker hosts

Exploitable max kernel version:

- Ubuntu 16.04:
  - Kernel 4.4.0-91.114
- Debian Stretch (namespace disable by default)
  - Kernel 4.9.30-2+deb9u5
- Redhat 7:
  - Kernel rt-3.10.0-693.5.2.rt56.626el





QUESTIONS ?