On the Effectiveness of Kernel Debloating via Compile-time Configuration

Mansour Alharthi, Hong Hu, Hyungon Moon, Taesoo Kim
The problem of bloated software

• High complexity: more vulnerabilities
• Unused interfaces: an attacker can use
• Unused code: more ROP gadget
Linux kernel is bloated

• Driving a variety of devices from servers to embedded
  • Server-friendly features
  • Embedded-only features

• Keep adopting new features
  • Support for new hardware
  • Performance optimizations
Problem of bloated kernel: avoidable bugs

• Linux distributions conservatively enable many features
  • Just in case a user wants them

• A system ends up suffering from a bug (vulnerability) in a feature that it never uses
  • which we should avoid
Example: X32 ABI

• Use x86_64 ISA: more registers than i386 (IA-32).
• Keep pointer size 32-bit: smaller memory footprints.
• Rarely used but enabled by default by popular distributions.
  • OpenSuse, Ubuntu, Solus.
• Related to a security-critical bug: CVE-2014-0038.
  • Local privilege escalation.
Example: CVE-2014-0038

• x32 ABI uses `compat_sys_recmmsg` to implement `recvmsg`.
• Incorrect casting at line 7 enables arbitrary memory write.
• Only the kernels that `CONFIG_X86_X32` enabled is vulnerable.

```
asmlinkage long compat_sys_recmmsg(int fd, struct compat_mmsghdr __user *mmsg, 
                                    unsigned int vlen, unsigned int flags, 
                                    struct compat_timespec __user *timeout) 
{

    //...
    if (COMPAT_USE_64BIT_TIME)
        return __sys_recmmsg(fd, (struct mmsghdr __user *)mmsg, vlen, 
                              flags | MSG_CMSG_COMPAT, 
                              (struct timespec *)timeout); /* bug here!!*/
```
Background: Linux kernel config. system

• Configuration options
  • E.g., CONFIG_NET, CONFIG_X86_X32
  • Determine if each source file/line is compile or not

• Configuration: a list of configuration options with the values

```plaintext
... 
CONFIG_X86_X32=y 
CONFIG_COMPAT_32=y 
CONFIG_COMPAT=y 
CONFIG_COMPAT_FOR_U64_ALIGNMENT=y 
CONFIG_SYSVIPC_COMPAT=y 
CONFIG_X86_DEV_DMA_OPS=y 
CONFIG_NET=y 
CONFIG_COMPAT_NETLINK_MESSAGES=y 
... 
```
Research goal

• The vulnerability-configuration option dependency

CONFIG_X86_X32

CVE-2014-0038

• Potential effectiveness of configuration option-grained tuning

# CVEs

# Enabled options

Default configuration

Deloaded configuration
Summary of results

• Dependency
  • $\exists$ options that many vulnerabilities depend on.
  • $\exists$ many options that at least one vulnerability depends on.

• Tuning
  • Popular programs do not need many options.
  • Disabling inessential options make the kernel less likely to have vulnerabilities.
Rest of this talk

• Dependency
  • Collecting the kernel vulnerabilities.
  • Locating the patches.
  • From a patch to the dependency.

• Tuning
  • Indirect study with existing configurations.
  • Direct study with hand-crafted configurations.

• Conclusion
Collecting the kernel vulnerabilities

• CVE data from National Vulnerability Database (NVD).
  • De facto standard, since 1999
  • Vulnerabilities found 2005 or after.
    • For easy access to patch: when the git was out
  • Only the upstream vulnerabilities.
    • For fair comparison between different distributions or forks
    • E.g., Ubuntu, Fedora or Android

→ 2046
→ 1773
→ 1530 vulnerabilities collected
Locating the patches from NVD entries

• The NVD entry for CVE-2014-0038

```
"cve" : {
  "data_type" : "CVE",
  "data_format" : "MITRE",
  "data_version" : "4.0",
  "CVE_data_meta" : {
    "ID" : "CVE-2014-0038",
    "ASSIGNER" : "cve@mitre.org"
  },
...
  "url" : "https://github.com/torvalds/linux/commit/2def2ef2ae5f3990aabdbe8a755911902707d268"
}
```

→ Located patches for 1242 entries
A patch example

```c
+++ b/net/compat.c
asmlinkage long compat_sys_recvmmsg(int fd, struct compat_mmsghdr __user *mmsg,
   -     if (COMPAT_USE_64BIT_TIME)
   -         return __sys_recvmmsg(fd, (struct mmsghdr __user *)mmsg, vlen,
   -             flags | MSG_CMSG_COMPAT,
   -             (struct timespec *) timeout);
   
   if (timeout == NULL)
     return __sys_recvmmsg(fd, (struct mmsghdr __user *)mmsg, vlen,
             flags | MSG_CMSG_COMPAT, NULL);
   
   -     if (get_compat_timespec(&ktspec, timeout))
+    if (compat_get_timespec(&ktspec, timeout))
         return -EFAULT;
         datagrams = __sys_recvmmsg(fd, (struct mmsghdr __user *)mmsg, vlen,
             flags | MSG_CMSG_COMPAT, &ktspec);
   
   -     if (datagrams > 0 && put_compat_timespec(&ktspec, timeout))
+    if (datagrams > 0 && compat_put_timespec(&ktspec, timeout))
         datagrams = -EFAULT;
```
From a patch to the dependencies (1)

• Find the options that determines if the patched lines are compiled

• Assumption: no change required $\Rightarrow$ no bug
From a patch to the dependencies (2)

• Kernel Makefiles determine if each file is included or not

Patch for CVE-2014-0038

net/compat.c:783
net/compat.c:792
net/compat.c:797

Linux Kernel Source Tree

CONFIG_COMPAT

net/compat.c
net/compat.c
net/compat.c

net/compat.c:797
net/compat.c:792
net/compat.c:783

CVE-2014-0038

net/compat.c

obj-$(CONFIG_NET):= socket.o
obj-$(CONFIG_COMPAT):= compat.o

obj-$(CONFIG_NET)+:= $(tmp-y)

# LLC has to be linked before the files in net/802/

obj-$(CONFIG_LLC):= llc/
obj-$(CONFIG_NET):= ethernet/802/sched/netlink/
From a patch to the dependencies (3)

- Kernel source

Patch for CVE-2

- net/core(skbuff)
- net/core(skbuff)
- net/core(skbuff)
- net/socket.c
- Include/linux/errqueue.h

Linux Kernel Source Tree

- net/
  - core/
  - skbuff.c
  - skbuff.c
- socket.c
- linux/
  - include/
  - errqueue.h

- lines 3805 & 3872
- line 3870 & 3872
- line 709
- line 22

fig options.
Observations from the graphs

• Case 1 (e.g., CVE-2014-0038):
  • Disabling one or more option completely discards all patches line.

• Case 2 (e.g. CVE-2017-7077):
  • There exists a patched line that is never discarded.
Inferring the number of active vulnerability

• Optimistic:
  • Discarding any of the patched line deactivates the vulnerability.
  • “OR” operation when inferring the numbers

• Conservative:
  • We must discard all patched lines to deactivate the vulnerability.
  • “AND” operation when inferring the numbers
Some numbers from the dependency study

• ∃Potentially *large* configuration options which are related to many vulnerabilities.
  • CONFIG_NET: 100, CONFIG_KVM: 46, CONFIG_PCI: 39

• Many *(701)* configuration options are related to at least one.

• Only **136 (11%)** vulnerabilities have a “bypass”.
  • Which debloating cannot deactivate in the worst case.
Can we then tune?

- Indirect study with existing configurations
  - Collected 66 default configurations

- Direct study with manual debloating
  - Created 2 minimal, application-specific configurations
More enabled options → more vulnerabilities
Manual debloating

• Minimal web server: nginx
  • Started from Ubuntu for x86
  • Correctness: if it serves a simple web page

• Minimal sensor node: mosquitto
  • Started from Buildroot for aarch64
  • Correctness: if a client can deliver a message to a server
Targeted debloating is effective

<table>
<thead>
<tr>
<th>Target</th>
<th>Distribution</th>
<th># Options</th>
<th># Bugs</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>nginx</td>
<td>Ubuntu</td>
<td>7598 → 1038 (86.3%)</td>
<td>929 → 234 (74.8%)</td>
<td>OR (Optimistic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1000 → 412 (58.8%)</td>
<td>AND</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1006 → 472 (53.1%)</td>
<td>AND with Bypasses (Conservative)</td>
</tr>
<tr>
<td>mosquitto</td>
<td>Buildroot</td>
<td>1229 → 581 (52.7%)</td>
<td>281 → 159 (43.4%)</td>
<td>OR (Optimistic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>472 → 265 (43.9%)</td>
<td>AND</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>526 → 347 (34.0%)</td>
<td>AND with Bypasses (Conservative)</td>
</tr>
</tbody>
</table>
Conclusion

• Most (89%) of vulnerabilities can be nullified by configuration.

• Application-specific debloating is effective (34-74% reduction).

• Next steps
  • Splitting large config options (e.g., CONFIG_NET)
  • Automating the configuration-grained debloating
Thank you!

Questions?