Systems Hardening Seminar (SHS)

SHS - Season 0, Pilot

Marius Momeu, Christopher Roemheld
Chair of IT Security (I20)

Wintersemester 2020/2021
13 July 2020
Who are we?

- self-proclaimed virtsec\(^1\) team within I20
- we evaluate CPU virtualization extensions (e.g. Intel VT-x, ARM, \ldots) in the context of security
- mitigate code-reuse attacks, data-oriented attacks\(^2\), heap misuses, \ldots
- combat split-personality malware\(^3\) \ldots
- we like to play CTFs in our spare time

---

\(^1\)virtualization security
\(^2\)”xMP: Selective Memory Protection for Kernel and User Space”
\(^3\)”Hiding in the Shadows: Empowering ARM for Stealthy Virtual Machine Introspection”
In this seminar, you are going to …

- … design defense mechanisms\textsuperscript{45} …

\textsuperscript{4} against memory corruption vulnerabilities
\textsuperscript{5} for Linux kernel- & userland software
In this seminar, you are going to ...

- ... design defense mechanisms\textsuperscript{4} ...
- ... and/or improve OS kernel/hypervisor fuzzing tools\textsuperscript{6} ...

\textsuperscript{4} against memory corruption vulnerabilities
\textsuperscript{5} for Linux kernel- & userland software
\textsuperscript{6} to find CVEs
In this seminar, you are going to …

- … design defense mechanisms\(^4\) …
- … and/or improve OS kernel/hypervisor fuzzing tools\(^5\) …
- … for x86 and/or ARM architectures.

\(^4\)against memory corruption vulnerabilities  
\(^5\)for Linux kernel- & userland software  
\(^6\)to find CVEs
In this seminar, you are going to...

- ... design defense mechanisms\(^4\) ...
- ... and/or improve OS kernel/hypervisor fuzzing tools\(^6\) ...
- ... for x86 and/or ARM architectures.
- And most importantly, ...

\(^4\) against memory corruption vulnerabilities
\(^5\) for Linux kernel- & userland software
\(^6\) to find CVEs
In this seminar, you are going to...

- ... design defense mechanisms\(^4\)...
- ... and/or improve OS kernel/hypervisor fuzzing tools\(^5\)...
- ... for x86 and/or ARM architectures.
- And most importantly, ...
- ... you are going to write a paper about your findings, ...

\(^4\) against memory corruption vulnerabilities
\(^5\) for Linux kernel- & userland software
\(^6\) to find CVEs
In this seminar, you are going to...

- ... design defense mechanisms\(^4\)...
- ... and/or improve OS kernel/hypervisor fuzzing tools\(^6\)...
- ... for x86 and/or ARM architectures.
- And most importantly, ...
- ... you are going to write a paper about your findings, ...
- ... and hold a presentation at the end of the semester.

\(^4\) against memory corruption vulnerabilities  
\(^5\) for Linux kernel- & userland software  
\(^6\) to find CVEs
Topics of Interest

- Mitigate **data-oriented attacks**
  - memory isolation, data-pointer integrity, information hiding
Topics of Interest

- Mitigate **data-oriented attacks**
  - memory isolation, data-pointer integrity, information hiding
- Mitigate **code-reuse attacks**
  - code isolation, control-flow integrity, code-pointer integrity
Topics of Interest

- Mitigate **data-oriented attacks**
  - memory isolation, data-pointer integrity, information hiding

- Mitigate **code-reuse attacks**
  - code isolation, control-flow integrity, code-pointer integrity

- **Heap hardening**
  - mitigate UAF, OOB, DF, IF bugs
Mitigate **data-oriented attacks**
- memory isolation, data-pointer integrity, information hiding

Mitigate **code-reuse attacks**
- code isolation, control-flow integrity, code-pointer integrity

**Heap hardening**
- mitigate UAF, OOB, DF, IF bugs

**Kernel & Hypervisor fuzzing**
- syzkaller, kAFL, Hyper-Cube, …
Topics of Interest

- Mitigate **data-oriented attacks**
  - memory isolation, data-pointer integrity, information hiding
- Mitigate **code-reuse attacks**
  - code isolation, control-flow integrity, code-pointer integrity
- **Heap hardening**
  - mitigate UAF, OOB, DF, IF bugs
- **Kernel & Hypervisor fuzzing**
  - syzkaller, kAFL, Hyper-Cube, …
- Explore **novel CPU features**
  - HLAT, SMAP, …
Topics of Interest

- Mitigate **data-oriented attacks**
  - memory isolation, data-pointer integrity, information hiding
- Mitigate **code-reuse attacks**
  - code isolation, control-flow integrity, code-pointer integrity
- **Heap hardening**
  - mitigate UAF, OOB, DF, IF bugs
- **Kernel & Hypervisor fuzzing**
  - syzkaller, kAFL, Hyper-Cube, …
- Explore **novel CPU features**
  - HLAT, SMAP, …
- **Virtual machine introspection**
  - stealthier VMI, faster VMI, …
Phase *nulla*: **Solve a remote qualification challenge** (details will follow) → 2 weeks

Phase I: **Pick a topic** → 1 week

Phase II: **Read literature** → 1 week

Phase III: **Write (first draft) / Prototype** → 6 weeks

Phase V: **Peer review** → 1 week

Phase VI: **Write (final paper) / Prototype** → 5 weeks

Phase VII: **Talk preparations** → 1 week

Phase VIII: **Extend your prototype into research (optional)** → *TBD*

---

Timeline is estimate; a detailed schedule will be announced before the start of WS2021
time & place: **Wednesdays, 10:00 - 12:00 (meeting biweekly)**, room 01.08.033

- …or online via **BBB** depending on how regulations develop

however, we will have **intermediate deliverables/presentations**

optional: irregular tutorials organized by us (scientific writing, interesting kernel/hypervisor CTF challenges, new processor features, …)

- **talks at the end** of the semester

- **10 student slots** (5 teams × 2) → **qualification challenge** (next slide)

- don’t forget to register via **the matching system until 21.07.2020!**

- target audience: **Master’s & Bachelor’s**

- language of instruction: **English**

- prerequisites: **operating systems, C & assembly** (x86 and/or ARM), Intel/ARM architecture specifics (useful), virtualization extensions (useful)
solve an easy CTF-like kernel/hypervisor challenge

goal: connect to the challenge server

and extract the secret (aka flag) from our underlying (Xen) hypervisor

send the obtained flag and your source code (in a .tar.gz) to momeu@sec.in.tum.de

deadline: 26th of July 23:59

successful candidates have priority in the seminar

---

8 PGP fingerprint: AD5A C550 7719 BD65 8165 0F39 D190 0A83 0CD1 3295
challenge hosted on praksrv.sec.in.tum.de → ssh user: gogaia, ssh pass: A3fdV7ZK7R

on each connection we spawn a minimalist Xen guest DomU where you have root access

underlying Xen hypervisor stores a 64-byte flag\(^9\) in one of its procedures

⇒ write a Linux kernel module that calls the Xen function to get you the flag

we provide the source code for the Xen hypervisor, DomU kernel, config files, and relevant server-side scripts

you may download the qualification challenge from our website\(^{10}\)!

\(^9\)flag template: flag{shs...}

\(^{10}\)https://www.sec.in.tum.de/i20/teaching/ws2020/systems-hardening
we provide the virtualization API in the Xen hypervisor on our x86 servers

for convenience, we will prepare for you an ARM server, with Xen deployed

(normally) you extend your kernel, and test it on our servers in a Xen guest\(^\text{11}\)

- if needed, we can extend Xen to assist your use-case

whatever (else) you need, we’re here for you, and we’ll find a solution

- simply drop to our office 01.08.057, or shoot us an email

we’ll keep in constant sync in-between the phases

\(^{11}\)aka domU - an unprivileged Xen guest VM
<table>
<thead>
<tr>
<th>Percentage</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>40%</td>
<td>Final Paper (Content, Style, Language, Scope, ... )</td>
</tr>
<tr>
<td>15%</td>
<td>Prototype / Design / Experiments</td>
</tr>
<tr>
<td>10%</td>
<td>Review</td>
</tr>
<tr>
<td>30%</td>
<td>Presentation (Content, Speaking, Style, Timeliness, ... )</td>
</tr>
<tr>
<td>5%</td>
<td>Discussion</td>
</tr>
<tr>
<td><strong>Σ</strong></td>
<td><strong>100%</strong> Total</td>
</tr>
</tbody>
</table>
Literature sources
- https://scholar.google.com
- https://semanticscholar.org
- https://dblp.uni-trier.de
- https://arxiv.org

Get around paywalls using https://www-ub-tum-de.eaccess.ub.tum.de/datenbanken

Researchers’ homepages can be valuable!
- source code, raw data, instructions, technical information, ...
Questions?

momeu@sec.in.tum.de
PGP fingerprint: AD5A C550 7719 BD65 8165 0F39 D190 0A83 0CD1 3295

@MariusMomeu

You may download the qualification challenge from our website\textsuperscript{12}!

\textsuperscript{12}https://www.sec.in.tum.de/i20/teaching/ws2020/systems-hardening