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, ...) in the context of security
- mitigate **code-reuse attacks, data-oriented attacks**[2]**, heap misuses,** ...
- combat **split-personality malware**[3] ...
- 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**[4][5] …

[4]against **memory corruption vulnerabilities**
[5]for **Linux kernel- & userland software**

In this seminar, you are going to …

- … design **defense mechanisms**[4][5] …
- … and/or improve **OS kernel/hypervisor fuzzing** tools[6] …

[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][5] …
- … and/or improve **OS kernel/hypervisor fuzzing** tools[6] …
- … 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][5] …
- … 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][5] …
- … 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, …

[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][5] …
- … 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

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, . . .

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, . . .

Structure

- Phase *nulla*: **Solve a remote qualification challenge** (details will follow) → 2 weeks
- Phase I: **Pick a topic** → 1 week[7]
- 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*

[7] Timeline is estimate; a detailed schedule will be announced before the start of WS2021

Orga

- 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 $\times$ 2) $\Longrightarrow$ **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)

Qualification challenge - overview

- 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 obtianed flag and your source code** (in a **.tar.gz**) to momeu@sec.in.tum.de[8]
- 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

Qualification challenge - detail

- 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**
- $\Longrightarrow$ **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

Seminar Infrastructure

- 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[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

Grading

	40 %	Final Paper (Content, Style, Language, Scope, . . .)
	15 %	Prototype / Design / Experiments
	10 %	Review
	30 %	Presentation (Content, Speaking, Style, Timeliness, . . .)
	5 %	Discussion
Σ	100 %	Total

Material

- 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, ...

The End (for now)

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[12]!

[12] https://www.sec.in.tum.de/i20/teaching/ws2020/systems-hardening