Final Paper Topics
"Operating Systems: Design and Security"

Paul Irofti

Rules

- 4 pages two columns paper (paper template in LaTeX here)
- students can work in teams of three
- a member of the team is a corresponding member
- the corresponding member will upload the final article for evaluation
- two teams cannot present the same topic
- choosing the topic is done by filling a table that will be sent before the end of the year
- the proposed topics are divided into four categories A,B,C,D

A. Tools

A1) Meltdown & Spectre attacks
  - choose a series of attacks
  - write or identify programs that implement them
  - create or identify multiple victim configurations
  - compare attacks on multiple vulnerable victims

A2) Test and compare several programs to identify and create ROP or `return-to-libc` attacks
  - test and showcase the produced gadgets
  - compare efficiency between programs
  - find cases where it does not work correctly
  - test on multiple programs

A3) write a driver for a very simple device (eg button, LED, etc.) where you bring the device's memory into the system's main memory and program the device's registers to do something; in the article describe the hardware component and the integration in the context of the operating system

A4) Implement algorithms for launching a `return-to-libc` attack

A5) Implement the gadget finding algorithms in a program

B. Research

For this type of assignment, you must create an article in which you present related ideas or the continuation of ideas presented in class. For example after Meltdown and Spectre, what other attacks have emerged from the same family? Include technical details and link to articles studied in the course.

Next to each topic you have a list of papers to choose from. This list is not exhaustive. If you find others, write to me and I will confirm if you can cover them in your paper.

B1) Virtual machines – implementation details from an OS perspective (paging, IO, access to physical devices, etc.)

- B1.1) FreeBSD Jails
  Poul-Henning Kamp and Robert NM Watson. Jails: Confining the omnipotent root. In *Proceedings of the 2nd International SANE Conference*, volume 43, page 116, 2000
- B1.2) Linux cgroups
- B1.3) OpenBSD vmm(4) (slides, video)

B2) Timing attacks

- B2.1) extensions
  - Daniel J Bernstein. Cache-timing attacks on AES. 2005
  - Yuval Yarom, Daniel Genkin, and Nadia Heninger. Cachebleed: a timing attack on openssl constant-time rsa. *Journal of Cryptographic Engineering*, 7(2):99–112, 2017
- B2.2) remote attacks
  - David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005
  - Billy Bob Brumley and Nicola Tuveri. Remote timing attacks are still practical. In *European Symposium on Research in Computer Security*, pages 355–371. Springer, 2011

B3) Cache attacks: Meltdown & Spectre

- B3.1) SGX attacks
  - Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure:{SGX} cache attacks are practical. In *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017
  - Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. Cache attacks on intel sgx. In *Proceedings of the 10th European Workshop on Systems Security*, pages 1–6, 2017
  - Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Malware guard extension: Using sgx to conceal cache attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer, 2017
- B3.2) Hyperthreading attacks

- Colin Percival. Cache missing for fun and profit, 2005
- Ben Gras, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. Translation leak-aside buffer: Defeating cache side-channel protections with {TLB} attacks. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 955–972, 2018

B3.3) Foreshadow attack

- Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 991–1008, 2018
- Ofir Weisse, Jo Van Bulck, Marina Minkin, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Raoul Strackx, Thomas F Wenisch, and Yuval Yarom. Foreshadow-ng: Breaking the virtual memory abstraction with transient out-of-order execution. 2018

B3.4) extensions

- Claudio Canella, Daniel Genkin, Lukas Giner, Daniel Gruss, Moritz Lipp, Marina Minkin, Daniel Moghimi, Frank Piessens, Michael Schwarz, Berk Sunar, et al. Fallout: Leaking data on meltdown-resistant cpus. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 769–784, 2019
- Stephan Van Schaik, Alyssa Milburn, Sebastian Österlund, Pietro Frigo, Giorgi Maisuradze, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. Ridl: Rogue in-flight data load. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 88–105. IEEE, 2019

B4) Rowhammer attacks

B4.1) remote attacks

- Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Rowhammer. js: A remote software-induced fault attack in javascript. In *International conference on detection of intrusions and malware, and vulnerability assessment*, pages 300–321. Springer, 2016
- Andrei Tatar, Radhesh Krishnan Konoth, Elias Athanasopoulos, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Throwhammer: Rowhammer attacks over the network and defenses. In *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, pages 213–226, 2018

B4.2) extensions

- Victor Van Der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clémentine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. Drammer: Deterministic rowhammer attacks on mobile platforms. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1675–1689, 2016
- Rui Qiao and Mark Seaborn. A new approach for rowhammer attacks. In *2016 IEEE international symposium on hardware oriented security and trust (HOST)*, pages 161–166. IEEE, 2016

B5) Buffer Overflow

- Autore Anonimo. Once upon a free ().. *Phrack Magazine*, 11(57), 2001
- blexim. Basic integer overflows. *Phrack Magazine*, 10(69), 2002
- Dark Spyrit and AB Jack. Win32 buffer overflows (location, exploitation, and prevention). *Phrack magazine*, 9:55, 1999

B6) Return-to-libc Attacks

- Sebastian Krahmer. x86-64 buffer overflow exploits and the borrowed code chunks exploitation technique, 2005
- Rafal Wojtczuk. The advanced return-into-lib (c) exploits: Pax case study. *Phrack Magazine, Volume 0x0b, Issue 0x3a, Phile# 0x04 of 0x0e*, 2001
- Ana Nora Sovarel, David Evans, and Nathanael Paul. Where's the feeb? the effectiveness of instruction set randomization. In *USENIX Security Symposium*, volume 10, 2005

B7) Address Space Layout Randomization

B7.1) design: W∧X, DEP, and PAX

B7.2) attacks (choose a subset)

- Hovav Shacham, Matthew Page, Ben Pfaff, Eu-Jin Goh, Nagendra Modadugu, and Dan Boneh. On the effectiveness of address-space randomization. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 298–307, 2004
- Tilo Müller. Aslr smack & laugh reference. In *Seminar on Advanced Exploitation Techniques*, 2008
- Yeongjin Jang, Sangho Lee, and Taesoo Kim. Breaking kernel address space layout randomization with intel tsx. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 380–392, 2016
- Ben Gras, Kaveh Razavi, Erik Bosman, Herbert Bos, and Cristiano Giuffrida. Aslr on the line: Practical cache attacks on the mmu. In *NDSS*, volume 17, page 13, 2017
- Dmitry Evtyushkin, Dmitry Ponomarev, and Nael Abu-Ghazaleh. Jump over aslr: Attacking branch predictors to bypass aslr. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE, 2016
- Yeongjin Jang, Sangho Lee, and Taesoo Kim. Breaking kernel address space layout randomization with intel tsx. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 380–392, 2016

B8) Return Oriented Programming

B8.1) extensions

- Ryan Roemer, Erik Buchanan, Hovav Shacham, and Stefan Savage. Return-oriented programming: Systems, languages, and applications. *ACM Transactions on Information and System Security (TISSEC)*, 15(1):1–34, 2012

- Erik Buchanan, Ryan Roemer, Hovav Shacham, and Stefan Savage. When good instructions go bad: Generalizing return-oriented programming to risc. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 27–38, 2008

B8.2) defense

- Todd Mortimer. Removing rop gadgets from openbsd. *AsiaBSDCon 2019*, page 13, 2019
- Lucas Davi, Ahmad-Reza Sadeghi, and Marcel Winandy. Ropdefender: A detection tool to defend against return-oriented programming attacks. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 40–51, 2011
- Kaan Onarlioglu, Leyla Bilge, Andrea Lanzi, Davide Balzarotti, and Engin Kirda. G-free: defeating return-oriented programming through gadget-less binaries. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 49–58, 2010
- Ping Chen, Hai Xiao, Xiaobin Shen, Xinchun Yin, Bing Mao, and Li Xie. Drop: Detecting return-oriented programming malicious code. In *International Conference on Information Systems Security*, pages 163–177. Springer, 2009

B9) Functional Correctness and Security Proofs: seL4 and Genode

C. Analysis of existing memory corruption vulnerabilities

For this topic, choose a known memory corruption vulnerability in a relatively popular program and analyze it. Memory corruption vulnerabilities include:

- Buffer Overflow (stack/heap/memory)
- Use-After-Free
- Out-of-Bounds Read/Write
- Double Free
- Integer Overflows/Underflows

To do this, you can roughly follow the following steps:

1. Search a program you would like to analyze on Mitre's CVE List. Some suggestions include browsers, the kernel, Nginx, Apache httpd, sudo, archivers (such as 7zip, tar), libc, sshd, busybox, FTP servers, Wireshark, PDF viewers, etc. You can pick any relatively popular program, even closed-source ones, but they might be harder.
2. Pick a CVE from the search results. Older CVEs have a higher chance of being easier to exploit than newer ones. Make sure it is a memory corruption vulnerability.
3. Find information about the bug. The less information there is online, the harder to analyze it is, but the more it is worth.

4. Acquire the vulnerable version of the software. Figure out how to build it and analyze it. Observe the patch for the bug. Discover where the bug is.

5. Write a report on the bug, explaining how it can be exploited, what the attacker can achieve (remote code execution, local privilege escalation, etc). Analyze the code and the flow of the user input. Relate the findings to the concepts taught in the course. Explain an exploitation strategy using ideas from the laboratory.

6. *[Optional]* Write an automated exploit for the bug, at least proving it exists. Preferably achieving a shell.

7. *[Optional]* Add any other interesting insights in the report. How could the bug have been avoided? How can such a bug be detected before it reaches release? How can the exploit for this bug be detected, locally or on the network?

D. Propose Your Own topic

Topic proposed by the student, related to the subject presented in the course – students can choose the topic only after receiving the teacher's approval. The proposal, accompanied by bibliographic references, must be sent to the address paul.irofti@fmi.unibuc.ro.