Lectures

05/07/2018 **0xd** `Project Presentations`

- Counterfeit Object-oriented Programming: On the Difficulty of Preventing Code Reuse Attacks in C++ Applications (http://ieeexplore.ieee.org/document/7163058/), IEEE S&P 2015 (optional).
- Enforcing Forward-Edge Control-Flow Integrity in GCC & LLVM (https://www.usenix.org/node/184460), USENIX Security 2015 (optional).
- Type Casting Verification: Stopping an Emerging Attack Vector (https://www.usenix.org/node/190956), USENIX Security 2015 (optional).
- A Tough `call`: Mitigating Advanced Code-Reuse Attacks at the Binary Level (http://ieeexplore.ieee.org/document/7546543/), IEEE S&P 2016 (optional).
- Drammer: Deterministic Rowhammer Attacks on Mobile Platforms (https://dl.acm.org/citation.cfm?id=2978406), ACM CCS 2016 (optional).
- Stack Bounds Protection with Low Fat Pointers (https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/stack-object-protection-low-fat-pointers/), NDSS 2017 (optional).
- Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing (https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/lee-sangho), USENIX Security 2017 (optional).
- CAn't Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory (https://www.usenix.org/node/203695), USENIX Security 2017 (optional).
- CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management (https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang), USENIX Security 2017 (optional).
- Delta Pointers: Buffer Overflow Checks Without the Checks (https://dl.acm.org/citation.cfm?id=3190553), EuroSys 2018 (optional).

04/30/2018 **0xc** `Special Topics`

- **Hacking Blind** (http://ieeexplore.ieee.org/document/6956567/), IEEE S&P 2014.
- **Framing Signals—A Return to Portable Shellcode** (http://ieeexplore.ieee.org/document/6956568/), IEEE S&P 2014.
- **The Devil is in the Constants: Bypassing Defenses in Browser JIT Engines** (https://www.ndss-symposium.org/ndss2015/ndss-2015-programme/devil-constants-bypassing-defenses-browser-jit-engines/), NDSS 2015.
- **Position-independent Code Reuse: On the Effectiveness of ASLR in the Absence of Information Disclosure** (https://www.cs.vu.nl/~herbertb/download/papers/pirop_eurosp18.pdf), IEEE EuroS&P 2018.
- Enabling Client-Side Crash-Resistance to Overcome Diversification and Information Hiding (http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/enabling-client-side-crash-resistance-overcome-diversification-information-hiding.pdf), NDSS 2016 (additional).
- Flip Feng Shui: Hammering a Needle in the Software Stack (https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/razavi), USENIX Security 2016 (additional).
- Oblivious Code Reuse: On the Effectiveness of Leakage Resilient Diversity (https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/address-oblivious-code-reuse-effectiveness-leakage-resilient-diversity/), NDSS 2017 (additional).
- Back To The Epilogue: Evading Control Flow Guard via Unaligned Targets (http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_05A-3_Biondo_paper.pdf), NDSS 2018 (additional).
- Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector (http://ieeexplore.ieee.org/document/7546546/), IEEE S&P 2016 (optional).
- Data-Oriented Programming: On the Expressiveness of Non-control Data Attacks (http://ieeexplore.ieee.org/document/7546545/), IEEE S&P 2016 (optional).
- Security Risks in Asynchronous Web Servers: When Performance Optimizations Amplify the Impact of Data-Oriented Attacks (http://www3.cs.stonybrook.edu/~mikepo/papers/asyncweb.eurosp18.pdf), IEEE EuroS&P 2018 (optional).

04/23/2018	**0xb**	`Kernel Security (cont'd)`

- **Prefetch Side-Channel Attacks: Bypassing SMAP and Kernel ASLR** (https://dl.acm.org/citation.cfm?id=2978356), ACM CCS 2016.
- **kR^X: Comprehensive Kernel Protection against Just-In-Time Code Reuse** (https://dl.acm.org/citation.cfm?id=3064216), EuroSys 2017.
- Enforcing Kernel Security Invariants with Data Flow Integrity (http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/enforcing-kernal-security-invariants-data-flow-integrity.pdf), NDSS 2016 (additional).
- Breaking Kernel Address Space Layout Randomization with Intel TSX (https://dl.acm.org/citation.cfm?id=2978321), ACM CCS 2016 (additional).
- UniSan: Proactive Kernel Memory Initialization to Eliminate Data Leakages (https://dl.acm.org/citation.cfm?id=2978366), ACM CCS 2016 (additional).
- How Double-Fetch Situations turn into Double-Fetch Vulnerabilities: A Study of Double Fetches in the Linux Kernel (https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-pengfei), USENIX Security 2017 (additional).
- Unleashing Use-Before-Initialization Vulnerabilities in the Linux Kernel Using Targeted Stack Spraying (https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/unleashing-use-initialization-vulnerabilities-linux-kernel-using-targeted-stack-spraying/), NDSS 2017 (optional).
- Secure Page Fusion with VUsion (https://dl.acm.org/citation.cfm?doid=3132747.3132781), ACM SOSP 2017 (optional).

04/16/2018	**0xa**	`Information Hiding`

- **Poking Holes in Information Hiding** (https://www.usenix.org/node/197249), USENIX Security 2016.
- **Undermining Information Hiding (and What to Do about It)** (https://www.usenix.org/node/197164), USENIX Security 2016.
- On the Effectiveness of Address-Space Randomization (https://dl.acm.org/citation.cfm?id=1030124), ACM CCS 2004 (additional).
- ASLR on the Line: Practical Cache Attacks on the MMU (https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/aslrcache-practical-cache-attacks-mmu/), NDSS 2017 (additional).
- No Need to Hide: Protecting Safe Regions on Commodity Hardware (https://dl.acm.org/citation.cfm?id=3064217), EuroSys 2017 (additional).
- From Zygote to Morula: Fortifying Weakened ASLR on Android (http://ieeexplore.ieee.org/document/6956579/), IEEE S&P 2014 (optional).
- CAIN: Silently Breaking ASLR in the Cloud (https://www.usenix.org/node/191961), USENIX WOOT 2015 (optional).

04/09/2018	**0x9**	`Code-Pointer Integrity`

- **Code-Pointer Integrity** (https://www.usenix.org/node/186160), USENIX OSDI 2014.
- **Missing the Point(er): On the Effectiveness of Code Pointer Integrity** (http://ieeexplore.ieee.org/document/7163060/), IEEE S&P 2016.
- **ASLR-Guard: Stopping Address Space Leakage for Code Reuse Attacks** (https://dl.acm.org/citation.cfm?id=2813694), ACM CCS 2015.
- Stack Object Protection with Low Fat Pointers (https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/stack-object-protection-low-fat-pointers/), NDSS 2017 (additional).
- PointGuard$^{\text{TM}}$: Protecting Pointers from Buffer Overflow Vulnerabilities (https://www.usenix.org/conference/12th-usenix-security-symposium/pointguard%E2%84%A2-protecting-pointers-buffer-overflow), USENIX Security 2003 (optional).

04/02/2018 **0x8** `Live ASLR`

- **Enhanced Operating System Security Through Efficient and Fine-grained Address Space Randomization** (https://www.usenix.org/node/180231), USENIX Security 2012.
- **Timely Rerandomization for Mitigating Memory Disclosures** (https://dl.acm.org/citation.cfm?id=2813691), ACM CCS 2015.
- **Shuffler: Fast and Deployable Continuous Code Re-Randomization** (https://www.usenix.org/node/199297), USENIX OSDI 2016.
- How to Make ASLR Win the Clone Wars: Runtime Re-Randomization (http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/how-make-aslr-win-clone-wars-runtime-re-randomization.pdf), NDSS 2016 (additional).
- CodeArmor: Virtualizing the Code Space to Counter Disclosure Attacks (http://ieeexplore.ieee.org/document/7962000/), IEEE EuroS&P 2017 (additional).
- Remix: On-demand Live Randomization (https://dl.acm.org/citation.cfm?id=2857705.2857726), ACM CODASPY 2015 (optional).

03/26/2018 **NUL** `Spring Recess`

- **No class**
- Breaking and Fixing Destructive Code Read Defenses (https://dl.acm.org/citation.cfm?id=3134626), ACSAC 2017 (optional).

03/19/2018 **0x7** `JIT-ROP Protection (cont'd)`

- **Heisenbyte: Thwarting Memory Disclosure Attacks using Destructive Code Reads** (https://dl.acm.org/citation.cfm?id=2813685), ACM CCS 2015.
- **Return to the Zombie Gadgets: Undermining Destructive Code Reads via Code Inference Attacks** (http://ieeexplore.ieee.org/document/7546544/), IEEE S&P 2016.
- Oxymoron: Making Fine-Grained Memory Randomization Practical by Allowing Code Sharing (https://www.usenix.org/node/184466), USENIX Security 2014 (additional).
- No-Execute-After-Read: Preventing Code Disclosure in Commodity Software (https://dl.acm.org/citation.cfm?id=2897891), ACM ASIACCS 2016 (additional).
- Isomeron: Code Randomization Resilient to (Just-In-Time) Return-Oriented Programming (https://www.ndss-symposium.org/ndss2015/ndss-2015-programme/isomeron-code-randomization-resilient-just-time-return-oriented-programming/), NDSS 2015 (optional).
- Defeating Zombie Gadgets by Re-randomizing Code upon Disclosure (https://link.springer.com/chapter/10.1007/978-3-319-62105-0_10), ESSoS 2017 (optional).

03/12/2018	**0x6**	`JIT-ROP Protection`

- **You Can Run but You Can't Read: Preventing Disclosure Exploits in Executable Code** (https://dl.acm.org/citation.cfm?id=2660378), ACM CCS 2014.
- **Readactor: Practical Code Randomization Resilient to Memory Disclosure** (http://ieeexplore.ieee.org/document/7163059/), IEEE S&P 2015.
- **Leakage-Resilient Layout Randomization for Mobile Devices** (http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/leakage-resilient-layout-randomization-mobile-devices.pdf), NDSS 2016.
- NORAX: Enabling Execute-Only Memory for COTS Binaries on AArch64 (http://ieeexplore.ieee.org/document/7958584/), IEEE S&P 2017 (additional).
- Protecting COTS Binaries from Disclosure-guided Code Reuse Attacks (https://dl.acm.org/citation.cfm?doid=3134600.3134634), ACSAC 2017 (additional).
- HideM: Protecting the Contents of Userspace Memory in the Face of Disclosure Vulnerabilities (https://dl.acm.org/citation.cfm?id=2699107), ACM CODASPY 2015 (optional).

03/05/2018	**0x5**	`Code Diversification`

- **Smashing the Gadgets: Hindering Return-Oriented Programming Using In-place Code Randomization** (http://ieeexplore.ieee.org/document/6234439/), IEEE S&P 2012.
- **Just-In-Time Code Reuse: On the Effectiveness of Fine-Grained Address Space Layout Randomization** (http://ieeexplore.ieee.org/document/6547134/), IEEE S&P 2013.
- **Compiler-assisted Code Randomization** (https://www.computer.org/csdl/proceedings/sp/2018/4353/00/435301a472-abs.html), IEEE S&P 2018.
- ILR: Where'd My Gadgets Go? (http://ieeexplore.ieee.org/document/6234437/), IEEE S&P 2012 (additional).
- Binary Stirring: Self-randomizing Instruction Addresses of Legacy x86 Binary Code (https://dl.acm.org/citation.cfm?id=2382216), ACM CCS 2012 (additional).
- Juggling the Gadgets: Binary-level Code Randomization using Instruction Displacement (https://dl.acm.org/citation.cfm?id=2897863), ACM ASIACCS 2016 (additional).
- Gadge Me If You Can: Secure and Efficient Ad-hoc Instruction-Level Randomization for x86 and ARM (https://dl.acm.org/citation.cfm?id=2484351), ACM ASIACCS 2013 (optional).
- SoK: Automated Software Diversity (http://ieeexplore.ieee.org/document/6956570/), IEEE S&P 2014 (optional).

02/26/2018	**0x4**	`Control-Flow Integrity (cont'd)`

- **Stitching the Gadgets: On the Ineffectiveness of Coarse-Grained Control-Flow Integrity Protection** (https://www.usenix.org/node/184482), USENIX Security 2014.
- **Control-Flow Bending: On the Effectiveness of Control-Flow Integrity** (https://www.usenix.org/node/190961), USENIX Security 2015.
- **The Dynamics of Innocent Flesh on the Bone: Code Reuse Ten Years Later** (https://dl.acm.org/citation.cfm?id=3134026), ACM CCS 2017.
- Out of Control: Overcoming Control-Flow Integrity (http://ieeexplore.ieee.org/document/6956588/), IEEE S&P 2014 (additional).
- Control Jujutsu: On the Weaknesses of Fine-Grained Control Flow Integrity (https://dl.acm.org/citation.cfm?id=2813646), ACM CCS 2015 (additional).
- ROP is Still Dangerous: Breaking Modern Defenses (https://www.usenix.org/node/184508), USENIX Security 2014 (optional).
- Size Does Matter: Why Using Gadget-Chain Length to Prevent Code-Reuse Attacks is Hard (https://www.usenix.org/node/184516), USENIX Security 2014 (optional).
- Losing Control: On the Effectiveness of Control-Flow Integrity under Stack Attacks (https://dl.acm.org/citation.cfm?id=2813671), ACM CCS 2015 (optional).

02/19/2018	**NUL**	`Presidents' Day`

- **No class**
- Control-Flow Integrity: Precision, Security, and Performance (https://arxiv.org/abs/1602.04056), arXiv.org (optional).

02/12/2018	**0x3**	`Control-Flow Integrity`

- **Control-Flow Integrity** (https://dl.acm.org/citation.cfm?id=1102165), ACM CCS 2005.
- **Control Flow Integrity for COTS Binaries** (https://www.usenix.org/node/174767), USENIX Security 2013.
- Practical Context-Sensitive CFI (https://dl.acm.org/citation.cfm?id=2813673), ACM CCS 2015 (additional).
- Per-Input Control-Flow Integrity (https://dl.acm.org/citation.cfm?id=2813644), ACM CCS 2015 (additional).
- Practical Control Flow Integrity and Randomization for Binary Executables (http://ieeexplore.ieee.org/document/6547133/), IEEE S&P 2013 (optional).
- Monitor Integrity Protection with Space Efficiency and Separate Compilation (https://dl.acm.org/citation.cfm?id=2516649), ACM CCS 2013 (optional).
- Opaque Control-Flow Integrity (https://www.ndss-symposium.org/ndss2015/ndss-2015-programme/opaque-control-flow-integrity/), NDSS 2015 (optional).
- CCFI: Cryptographically Enforced Control Flow Integrity (https://dl.acm.org/citation.cfm?id=2813676), ACM CCS 2015 (optional).
- Efficient Protection of Path-Sensitive Control Security (https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/ding), USENIX Security 2017 (optional).
- Venerable Variadic Vulnerabilities Vanquished (https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/biswas), USENIX Security 2017 (optional).
- Let's talk about CFI: clang edition (https://blog.trailofbits.com/2016/10/17/lets-talk-about-cfi-clang-edition/) | Microsoft Edition (https://blog.trailofbits.com/2016/12/27/lets-talk-about-cfi-microsoft-edition/), Trail of Bits (optional).

02/05/2018	**0x2**	`Kernel Security`

- **kGuard: Lightweight Kernel Protection against Return-to-user Attacks** (https://www.usenix.org/node/180230), USENIX Security 2012.
- **ret2dir: Rethinking Kernel Isolation** (https://www.usenix.org/node/184468), USENIX Security 2014.
- Practical Timing Side Channel Attacks against Kernel Space ASLR (http://ieeexplore.ieee.org/document/6547110/), IEEE S&P 2013 (additional).
- PT-Rand: Practical Mitigation of Data-only Attacks against Page Tables (https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/pt-rand-practical-mitigation-data-only-attacks-against-page-tables/), NDSS 2017 (additional).
- A Tale of Two Kernels: Towards Ending Kernel Hardening Wars with Split Kernel (https://dl.acm.org/citation.cfm?id=2660331), ACM CCS 2014 (optional).
- Nested Kernel: An Operating System Architecture for Intra-Kernel Privilege Separation (https://dl.acm.org/citation.cfm?id=2694386), ACM ASPLOS 2016 (optional).
- Jump over ASLR: Attacking Branch Predictors to Bypass ASLR (http://ieeexplore.ieee.org/document/7783743/), IEEE MICRO 2016 (optional).

01/29/2018	**0x1**	`Introduction | Basic Concepts`

- **Lecture slides** (slides/l01.pdf)
- **SoK: Eternal War in Memory** (http://ieeexplore.ieee.org/document/6547101/), IEEE S&P 2013.
- x86 Assembly (https://en.wikibooks.org/wiki/X86_Assembly) | x86-64 Assembly (https://software.intel.com/en-us/articles/introduction-to-x64-assembly)
- x86 Instruction Set Reference (http://kernfunny.org/x86/)
- Stack frame layout: x86 (https://eli.thegreenplace.net/2011/02/04/where-the-top-of-the-stack-is-on-x86/) | x86-64 (http://eli.thegreenplace.net/2011/09/06/stack-frame-layout-on-x86-64)
- Position Independent Code (PIC): x86 (https://eli.thegreenplace.net/2011/11/03/position-independent-code-pic-in-shared-libraries) | x86-64 (https://eli.thegreenplace.net/2011/11/11/position-independent-code-pic-in-shared-libraries-on-x64)
- Anatomy of a Program in Memory (http://duartes.org/gustavo/blog/post/anatomy-of-a-program-in-memory/)
- Linux x86 Program Start Up (http://dbp-consulting.com/tutorials/debugging/linuxProgramStartup.html)
- The ELF file format (http://www.gabriel.urdhr.fr/2015/09/28/elf-file-format/)

Meetings

Monday 3PM – 5:20PM (M hour)
CIT (Thomas J. Watson Sr. Center for Information Technology) (http://brown.edu/Facilities/Facilities_Management/maps/index.php#building/WATSONCIT) 477 (Lubrano (https://cs.brown.edu/about/rooms/))

Instructor

`Vasileios (Vasilis) Kemerlis`

`https://cs.brown.edu/~vpk` (https://cs.brown.edu/~vpk)
`echo @cs.brown.edu|sed 's/^/vpk/'`
CIT (Thomas J. Watson Sr. Center for Information Technology) (https://brown.edu/Facilities/Facilities_Management/maps/index.php#building/WATSONCIT) 505 (Mon. 6PM – 8PM)

Communication

`course.csci.2951u.2018-spring.s01@lists.brown.edu` (mailto:course.csci.2951u.2018-spring.s01@lists.brown.edu)

Announcements

05/07/2018	Project presentations.
04/23/2018	Lecture 0xc readings posted.
04/16/2018	Lecture 0xb readings posted.
04/09/2018	Lecture 0xa readings posted.
04/02/2018	Lecture 0x9 readings posted.
03/26/2018	No class today.
03/19/2018	Lecture 0x8 readings posted.
03/12/2018	Lecture 0x7 readings posted.
03/05/2018	Lecture 0x6 readings posted.
02/26/2018	Lecture 0x5 readings posted.
02/19/2018	No class today.
02/16/2018	Lecture 0x4 readings posted.
02/05/2018	Lecture 0x3 readings posted.
01/29/2018	Lecture 0x2 readings posted.
01/29/2018	Lecture 0x1 readings posted.
01/24/2018	Welcome to CSCI 2951U!