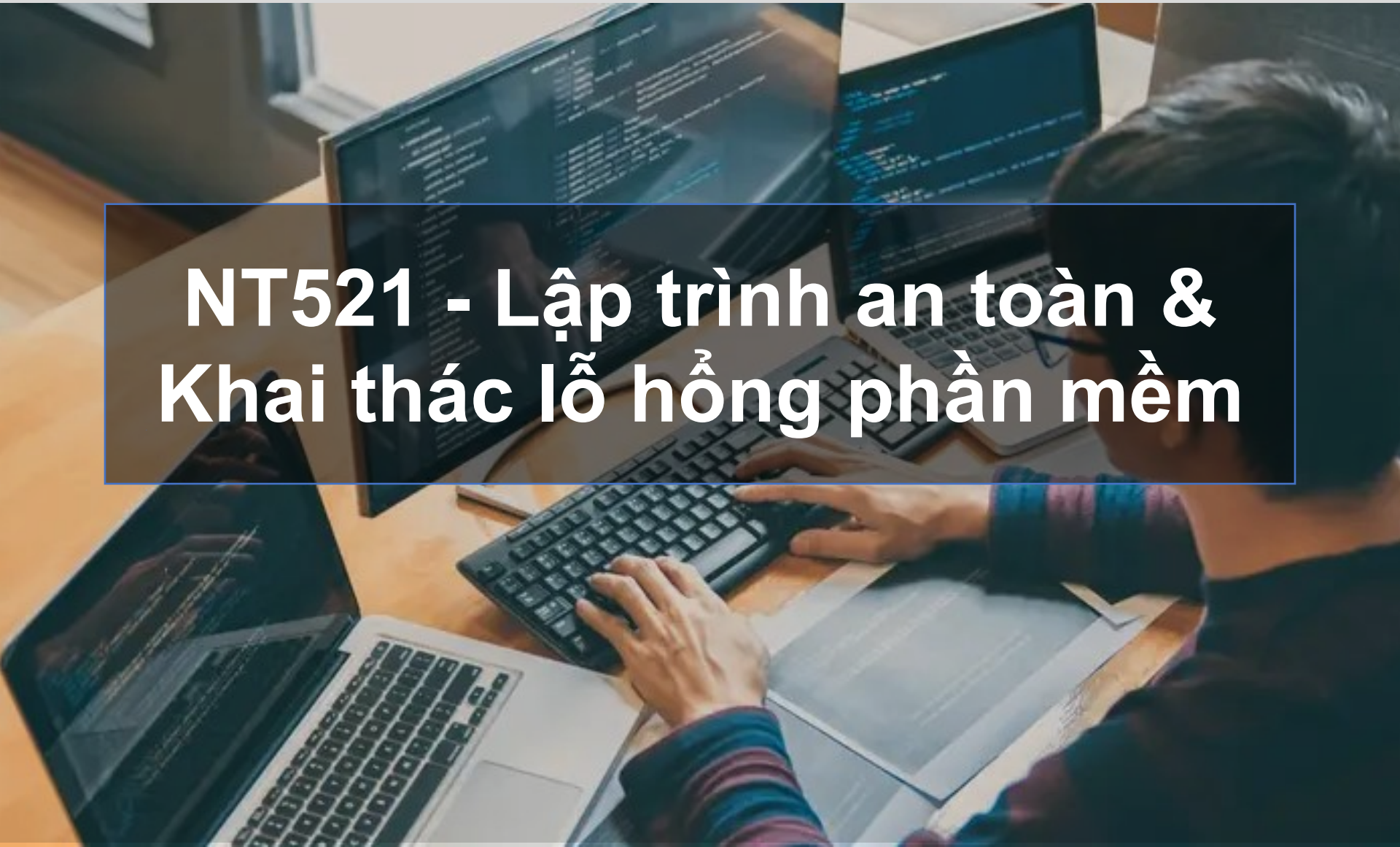


NT521 - Lập trình an toàn & Khai thác lỗ hổng phần mềm

Các hướng tương lai của Bảo mật và khai thác lỗ hổng phần mềm

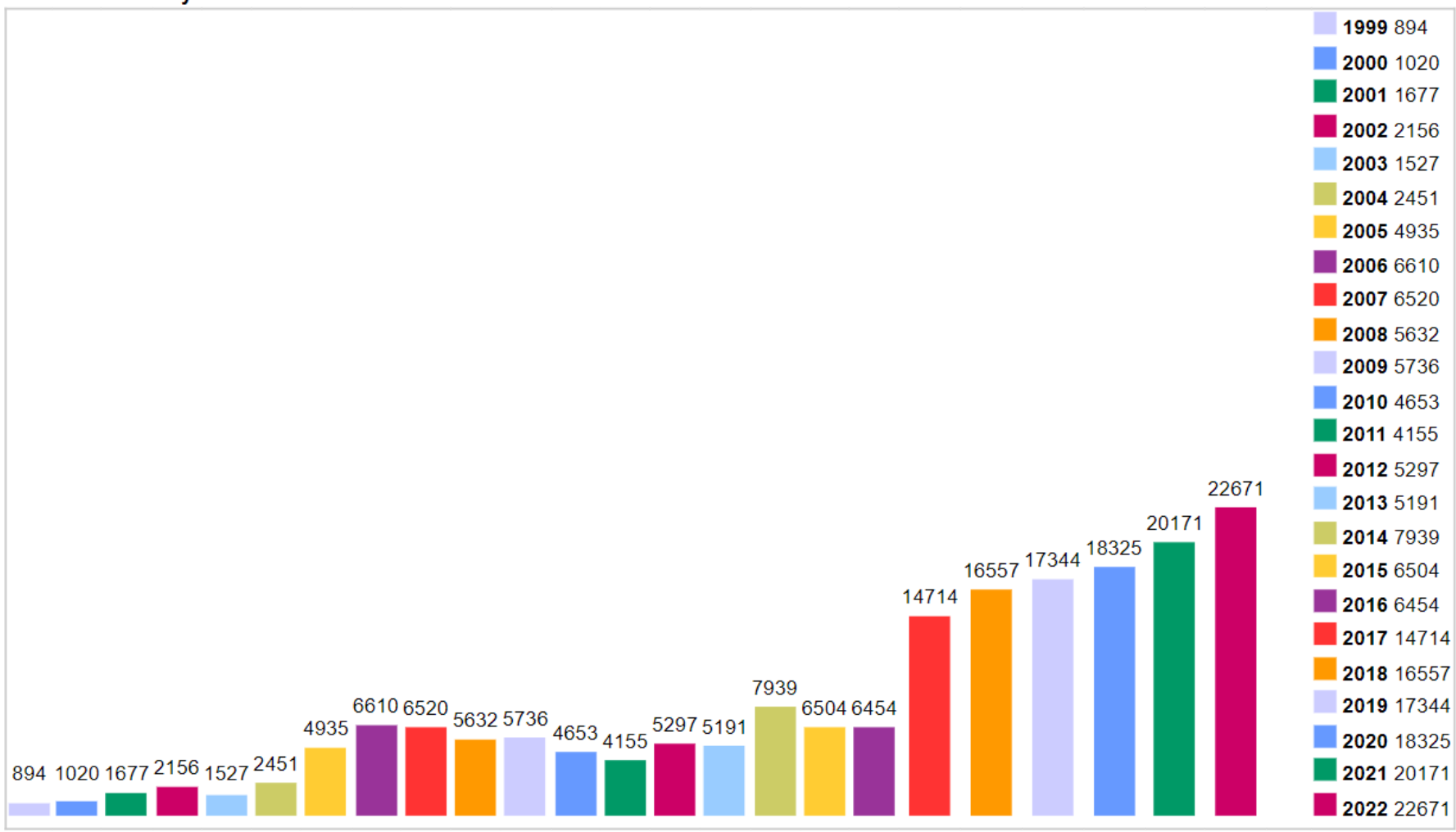
Nội dung

- Một số hướng nghiên cứu tương lai của lĩnh vực An toàn, bảo mật phần mềm (Software Security).
- Một số hướng nghiên cứu tương lai của lĩnh vực Khai thác Lỗ hổng phần mềm (Exploitation).

#1: Một số hướng nghiên cứu về An toàn, bảo mật phần mềm

The Future Direction of Software Security

Thống kê CVE qua các năm



https://www.cvedetails.com/browse-by-date.php

Thực trạng về an toàn phần mềm

- Các ứng dụng và các hệ thống sẽ **KHÔNG BAO GIỜ** an toàn theo cách hoàn hảo nhất.
  - Ứng dụng ngày càng phức tạp hơn
- Việc khai thác lỗ hổng ngày càng trở nên phức tạp hơn:
  - Nhiều lỗ hổng (bug) hơn
  - Nhiều thời gian (time) hơn
  - Nhiều tiền bạc (money) hơn

Thực trạng về an toàn phần mềm

- Các dạng *khai thác dựa trên kiểu phá vỡ cấu trúc bộ nhớ* (memory corruption) **sẽ trở nên khó thực hiện** hơn trong tương lai
  - Do các biện pháp bảo vệ của hệ điều hành
  - Chi phí để thực hiện xâu chuỗi lại các lỗ hổng trên phần mềm quá cao
  - Phức tạp trong hành động trên chương trình mục tiêu mà vẫn đảm bảo không kích hoạt cơ chế giám sát và phát hiện hành vi thao túng trên ứng dụng.
- Các khai thác dựa trên các *khiếm khuyết của cài đặt và xử lý luận lý (implementation & logic flaw)* sẽ luôn tồn tại.

Một số hướng nghiên cứu

- Phát triển các kỹ thuật làm rối (obfuscation) cho phần mềm

[1]Sebastian Schrittwieser, Stefan Katzenbeisser, Johannes Kinder, Georg Merzdovnik, and Edgar Weippl. 2016. **Protecting Software through Obfuscation: Can It Keep Pace with Progress in Code Analysis?** ACM Comput. Surv. 49, 1, Article 4 (March 2017), 37 pages. https://doi.org/10.1145/2886012

[2] Shohreh Hosseinzadeh, Sampsa Rauti, Samuel Laurén, Jari-Matti Mäkelä, Johannes Holvitie, Sami Hyrynsalmi, Ville Leppänen, "**Diversification and obfuscation techniques for software security: A systematic literature review**", Information and Software Technology, 2018.

[3] H. Wang, S. Wang, D. Xu, X. Zhang and X. Liu, "**Generating Effective Software Obfuscation Sequences With Reinforcement Learning**," in IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 3, pp. 1900-1917, 1 May-June 2022, doi: 10.1109/TDSC.2020.3041655.

[4] S. A. Ebad, A. A. Darem and J. H. Abawajy, "**Measuring Software Obfuscation Quality–A Systematic Literature Review**," in IEEE Access, vol. 9, pp. 99024-99038, 2021, doi: 10.1109/ACCESS.2021.3094517.

Một số hướng nghiên cứu

- Kỹ thuật **phát hiện sao chép mã nguồn/phát hiện sự tương đồng trong mã nhị phân của chương trình** (Software Clone Detection/ Binary Code Similarity Detection)
  - H. Zhang and K. Sakurai, "**A Survey of Software Clone Detection From Security Perspective**," in IEEE Access, vol. 9, pp. 48157-48173, 2021,
  - Donghai Tian, Xiaoqi Jia, Rui Ma, Shuke Liu, Wenjing Liu, Changzhen Hu: **BinDeep: A deep learning approach to binary code similarity detection**. Expert Syst. Appl. 168: 114348 (2021)
  - Geunwoo Kim, Sanghyun Hong, Michael Franz, and Dokyung Song. 2022. **Improving cross-platform binary analysis using representation learning via graph alignment**. In Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2022). Association for Computing Machinery, New York, NY, USA, 151–163.
  - Andrea Marcelli, Mariano Graziano, Xabier Ugarte-Pedrero, Yanick Fratantonio, Mohamad Mansouri, Davide Balzarotti. **How Machine Learning Is Solving the Binary Function Similarity Problem**. USENIX Security '22.
    - https://github.com/Cisco-Talos/binary_function_similarity

Một số hướng nghiên cứu

- Kỹ thuật **phát hiện sao chép mã nguồn/phát hiện sự tương đồng trong mã nhị phân của chương trình** (Software Clone Detection/ Binary Code Similarity Detection)
  - J. Yang, C. Fu, X. -Y. Liu, H. Yin and P. Zhou, "**Codee: A Tensor Embedding Scheme for Binary Code Search**," in IEEE Transactions on Software Engineering, vol. 48, no. 7, pp. 2224-2244, 1 July 2022, doi: 10.1109/TSE.2021.3056139.
    - Github: https://github.com/ycachy/Codee
  - Hao Wang, Wenjie Qu, Gilad Katz, Wenyu Zhu, Zeyu Gao, Han Qiu, Jianwei Zhuge, and Chao Zhang. 2022. **JTrans: jump-aware transformer for binary code similarity detection**. In Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2022). Association for Computing Machinery, New York, NY, USA, 1–13.
    - Github: https://github.com/vul337/jTrans
  - Asteria: Deep Learning-based AST-Encoding for Cross-platform Binary Code Similarity Detection, 2021.
    - Github: https://github.com/Asteria-BCSD/Asteria
  - Xem thêm: https://github.com/SystemSecurityStorm/Awesome-Binary-Similarity

#2: Một số hướng nghiên cứu về Khai thác lỗ hổng phần mềm

The Future Direction of Software Exploitation

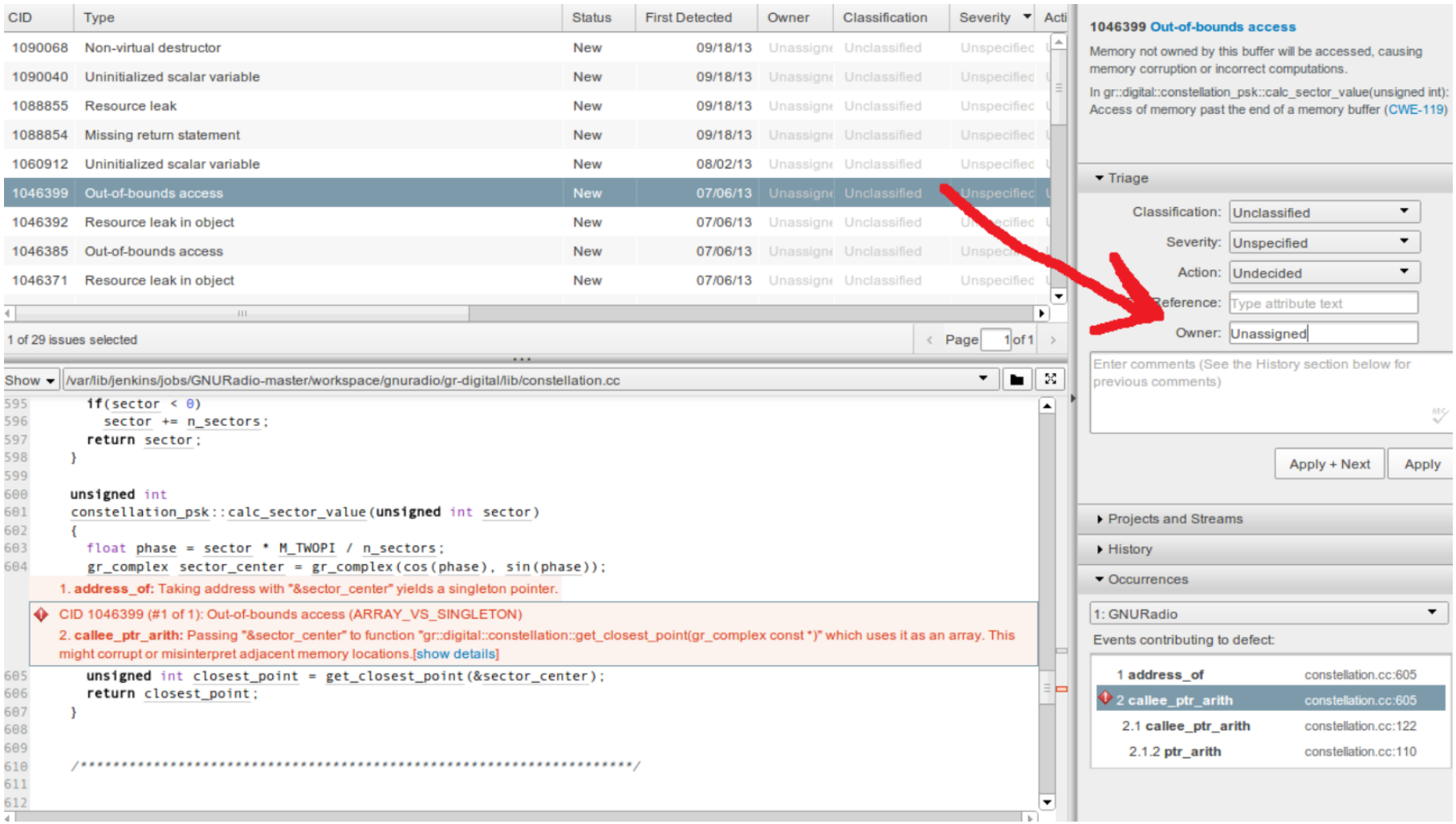
Khai thác lỗ hổng phần mềm

- Tốn hàng giờ/ngày/tuần/tháng cho việc tìm kiếm bug
  - **Bug Hunting:** Đây là tác vụ tốn nhiều thời gian khi phải tìm kiếm lỗ hổng trong các trường hợp có mã nguồn, và không có mã nguồn.
- Cách nào để tìm bug nhanh hơn?
  - **Tự động hóa (Automation)**

Static Code Analyzer

- Các trình phân tích mã nguồn tĩnh (Static Code Analyzer) có thể giúp tìm bug theo phương pháp tĩnh, tuy nhiên trong đa phần các trường hợp sẽ bỏ lỡ nhiều lỗ hổng
  - Rất khó để phát hiện các lỗ hổng UAF theo phương pháp phân tích tĩnh
- **Coverity** là một trong những công cụ phổ biến có thể dùng nâng cao chất lượng của quá trình phân tích tĩnh mã nguồn.

Static Code Analyzer: Coverity



CID	Type	Status	First Detected	Owner	Classification	Severity	Acti
1090068	Non-virtual destructor	New	09/18/13	Unassigne	Unclassified	Unspecified	
1090040	Uninitialized scalar variable	New	09/18/13	Unassigne	Unclassified	Unspecified	
1088855	Resource leak	New	09/18/13	Unassigne	Unclassified	Unspecified	
1088854	Missing return statement	New	09/18/13	Unassigne	Unclassified	Unspecified	
1060912	Uninitialized scalar variable	New	08/02/13	Unassigne	Unclassified	Unspecified	
1046399	Out-of-bounds access	New	07/06/13	Unassigne	Unclassified	Unspecified	
1046392	Resource leak in object	New	07/06/13	Unassigne	Unclassified	Unspecified	
1046385	Out-of-bounds access	New	07/06/13	Unassigne	Unclassified	Unspecified	
1046371	Resource leak in object	New	07/06/13	Unassigne	Unclassified	Unspecified	

1 of 29 issues selected

Page 1 of 1

Show /var/lib/jenkins/jobs/GNURadio-master/workspace/gnuradio/gr-digital/lib/constellation.cc

```
595       if(sector < 0)
596         sector += n_sectors;
597       return sector;
598     }
599
600     unsigned int
601     constellation_psk::calc_sector_value(unsigned int sector)
602     {
603       float phase = sector * M_TWOPI / n_sectors;
604       gr_complex sector_center = gr_complex(cos(phase), sin(phase));
```

1. **address_of:** Taking address with "&sector_center" yields a singleton pointer.

CID 1046399 (#1 of 1): Out-of-bounds access (ARRAY_VS_SINGLETON)

2. **callee_ptr_arith:** Passing "&sector_center" to function "gr::digital::constellation::get_closest_point(gr_complex const *)" which uses it as an array. This might corrupt or misinterpret adjacent memory locations.[show details]

```
605       unsigned int closest_point = get_closest_point(&sector_center);
606       return closest_point;
607     }
608
609
610     /*************************************************************************/
611
612
```

1046399 Out-of-bounds access

Memory not owned by this buffer will be accessed, causing memory corruption or incorrect computations.

In gr::digital::constellation_psk::calc_sector_value(unsigned int): Access of memory past the end of a memory buffer (CWE-119)

Triage

Classification: Unclassified

Severity: Unspecified

Action: Undecided

Reference: Type attribute text

Owner: Unassigned

Enter comments (See the History section below for previous comments)

Apply + Next | Apply

Projects and Streams

History

Occurrences

1: GNURadio

Events contributing to defect:

1 **address_of**	constellation.cc:605
2 **callee_ptr_arith**	constellation.cc:605
2.1 **callee_ptr_arith**	constellation.cc:122
2.1.2 **ptr_arith**	constellation.cc:110

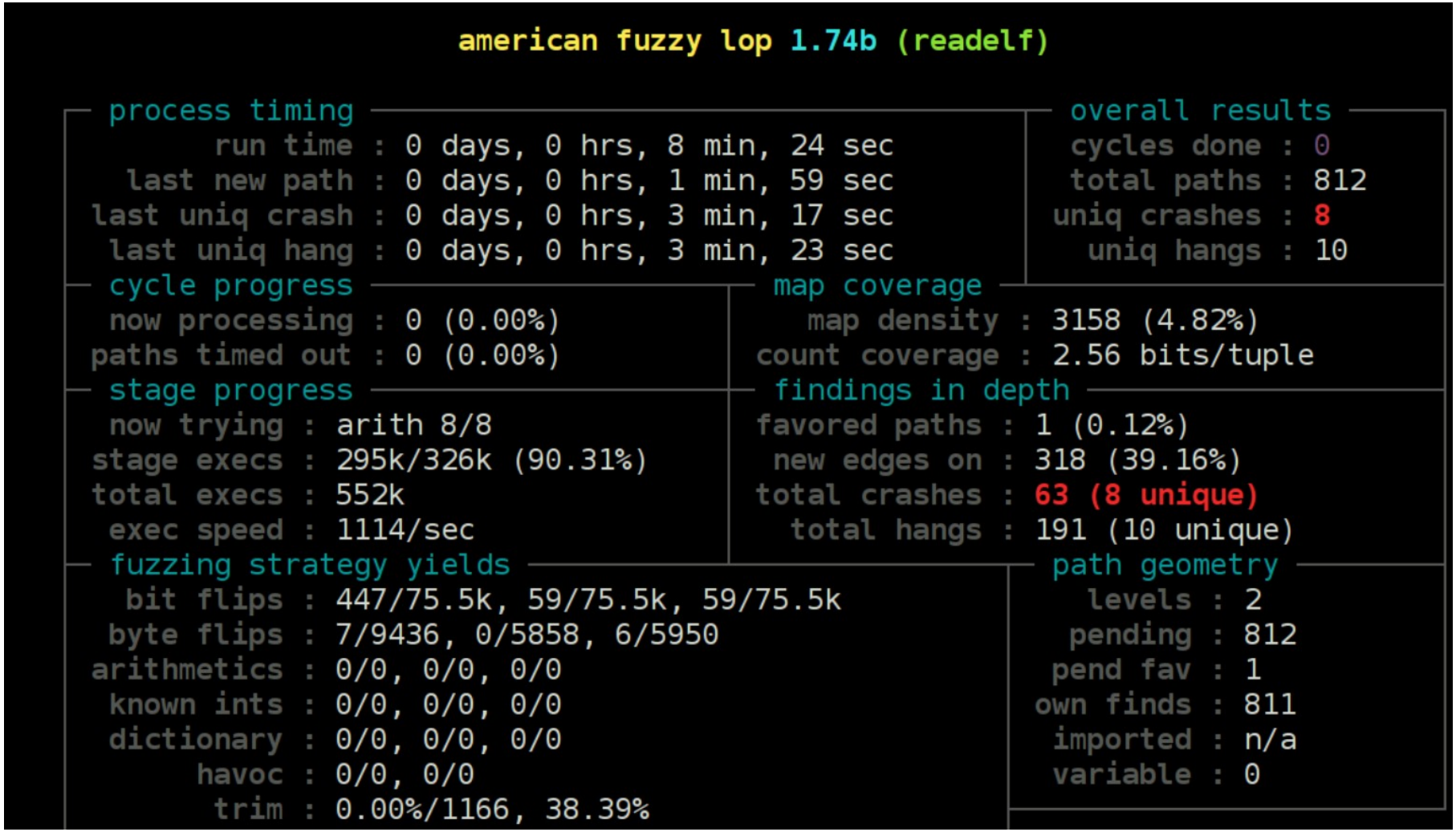
Fuzzing

- **Fuzzing** – kỹ thuật biến đổi dữ liệu và đẩy vào các chương trình mục tiêu để quan sát khả năng chương trình không có sự kiểm soát trong xử lý những dữ liệu này, dẫn tới chương trình bị crash.
- **Fuzzing** được xem gần như là nguồn gốc của hơn 95% của các loại lỗ hổng được tìm thấy trong khoảng 10 năm vừa qua.

American Fuzzy Lop (AFL)

- **AFL** là một trình fuzzer hướng an toàn thông tin (security-oriented), có chức năng chèn và sử dụng "instrumentation" – vốn được nó chèn tại thời điểm biên dịch của chương trình nhằm tìm lỗ hổng phần mềm.
  - Đòi hỏi mã nguồn của chương trình để đạt hiệu năng vượt trội
  - Vẫn có thể sử dụng trong các trường hợp không có mã nguồn, lấy thêm thông tin từ quá trình dịch ngược chương trình.

American Fuzzy Lop (AFL)



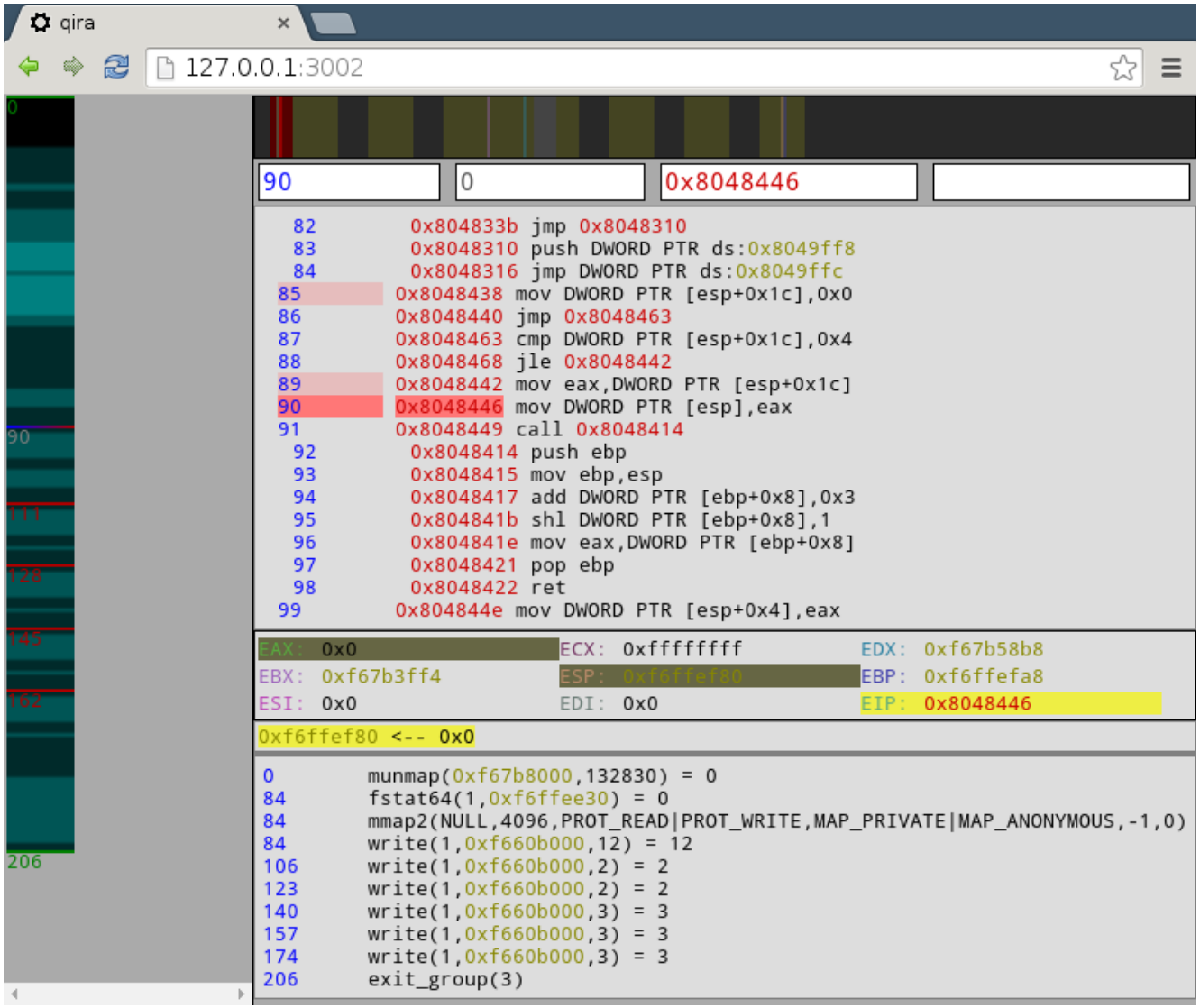
Đặc điểm cơ bản của các Bug hiện đại

- Khi các bug ngày càng trở nên có tính đặc biệt/chọn lọc và phức tạp hơn, các phương pháp fuzzing sẽ giúp tìm ra các bug này tốt hơn, nhanh hơn.
- Nhiều bug tìm thấy hiện tại được báo cáo rằng, **phải thực thi** bởi **một điều kiện rất đặc biệt** (very specific condition).

QIRA

- Timeless debugger:
  - Có thể quan sát chương trình tại bất kỳ điểm nào của quá trạng thái thực thi chương trình tương ứng với dữ liệu đầu vào cho trước
  - Có thể di chuyển **forward** và **backward**.
- Tham khảo: http://qira.me

QIRA


qira
127.0.0.1:3002
90
0
0x8048446
82 0x804833b jmp 0x8048310
83 0x8048310 push DWORD PTR ds:0x8049ff8
84 0x8048316 jmp DWORD PTR ds:0x8049ffc
85 0x8048438 mov DWORD PTR [esp+0x1c],0x0
86 0x8048440 jmp 0x8048463
87 0x8048463 cmp DWORD PTR [esp+0x1c],0x4
88 0x8048468 jle 0x8048442
89 0x8048442 mov eax,DWORD PTR [esp+0x1c]
90 0x8048446 mov DWORD PTR [esp],eax
91 0x8048449 call 0x8048414
92 0x8048414 push ebp
93 0x8048415 mov ebp,esp
94 0x8048417 add DWORD PTR [ebp+0x8],0x3
95 0x804841b shl DWORD PTR [ebp+0x8],1
96 0x804841e mov eax,DWORD PTR [ebp+0x8]
97 0x8048421 pop ebp
98 0x8048422 ret
99 0x804844e mov DWORD PTR [esp+0x4],eax
EAX: 0x0
ECX: 0xffffffff
EDX: 0xf67b58b8
EBX: 0xf67b3ff4
ESP: 0xf6ffef80
EBP: 0xf6ffefa8
ESI: 0x0
EDI: 0x0
EIP: 0x8048446
0xf6ffef80 <-- 0x0
0 munmap(0xf67b8000,132830) = 0
84 fstat64(1,0xf6ffee30) = 0
84 mmap2(NULL,4096,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
84 write(1,0xf660b000,12) = 12
106 write(1,0xf660b000,2) = 2
123 write(1,0xf660b000,2) = 2
140 write(1,0xf660b000,3) = 3
157 write(1,0xf660b000,3) = 3
174 write(1,0xf660b000,3) = 3
206 exit_group(3)
90
206

Các kỹ thuật nâng cao

- **Taint Analysis**
  - Theo vết (tracing) mức độ ảnh hưởng của dữ liệu đầu vào đối với chương trình nhị phân, theo dõi cách nó ảnh hưởng lên quá trình thực thi.
  - PANDA, QIRA.
- **Symbolic Execution + SAT/SMT Solving:**
  - Chứng minh rằng một điều kiện nhất định có thể tồn tại bên trong quá trình thực thi của chương trình để "tuyên bố" có sự tồn tại của một bug "khó".
  - Z3, SMT-LIB.
- **Học máy (Machine Learning)**

Tương lai của khai thác lỗ hổng

- Một ví dụ về dự **án Cyber Grand Challenge (CGC) của DARPA**
- Tìm phương pháp hoàn toàn tự động có khả năng:
  - Tìm kiếm lỗ hổng (hộp trắng và hộp đen)
  - Vá (patching) lỗ hổng được tìm thấy
  - Viết mã khai thác cho các lỗ hổng được tìm thấy

Một vài tổ chức tham gia thi đấu ở CGC

Berkeley
UNIVERSITY OF CALIFORNIA

TRAIL OF BITS

Raytheon
Technologies

A.Secure

SRI International®

MIT
LINCOLN
LABORATORY

Tương lai của khai thác lỗ hổng

- Hệ thống "**Cyber Reasoning System -CRS**" được phát triển trong lộ trình thi đấu của CGC, có khả năng tìm và khai thác lỗ hổng phần mềm hoàn toàn tự động.
- Kỹ thuật đằng sau hệ thống này là các trình fuzzer thông minh (smart fuzzing) được hướng dẫn/điều khiển bởi taint analysis, constraint solver, và ML.

T. Avgerinos et al., "The Mayhem Cyber Reasoning System," in IEEE Security & Privacy, vol. 16, no. 2, pp. 52-60, March/April 2018, doi: 10.1109/MSP.2018.1870873.

Mayhem CRS

The Mayhem Cyber Reasoning System

Thanassis Avgerinos, David Brumley, John Davis, Ryan Goulden, Tyler Nighswander, Alex Rebert, and Ned Williamson | ForAllSecure

Mayhem, one of the first generation of autonomous computer security bots that finds and fixes vulnerabilities without human intervention, won the DARPA Cyber Grand Challenge in August 2016. In this article, we detail Mayhem's creation and look forward to a future where autonomous bots will radically improve computer system security.

We are losing the battle against criminals who break into our computer systems. The reason: we rely only on humans to find new vulnerabilities and fix them. What if, however, we did not need to rely on human effort alone to find vulnerabilities? What if we could build intelligent computer bots that can find and fix vulnerabilities? And what if these intelligent bots could reduce the time to identify and remediate a vulnerability from human time scales—days, months, or years—to computer time scales of milliseconds?

The art of securing computer systems and processes against malicious actors has a broad frontier. DARPA's Cyber Grand Challenge (CGC) explored a new technology on the forefront of cybersecurity: the cyber reasoning system (CRS). A CRS is a fully autonomous system that takes complete responsibility for defending a set of software services. CRSs competing in the Cyber Grand Challenge demonstrated techniques in all core cybersecurity areas, including automatically developing firewall rules to stop attack traffic, analyzing programs to find bugs before an attacker, and patching vulnerabilities in compiled programs without any access to source code. Software defenses that might take human analysts hours, days, or weeks to develop and test were all deployed at machine speeds, on the order of tens of seconds, with no humans in the loop. This first generation of CRSs offers hope for an automatic first line of defense against attacks conducted using novel exploits on large scales at machine speeds, in addition to greatly tightening a defender's response time to other, more typical attacks.

We are the authors, designers, and developers of Mayhem, the winning CRS in the Cyber Grand Challenge, and this article discusses the design and implementation of Mayhem, lessons learned while preparing for the challenge, and our key takeaways from the competition.

DARPA's Cyber Grand Challenge

The CGC was a competitive, symmetric game played by seven fully autonomous CRSs and moderated by a "referee" scoring system. Here, we provide a brief overview of the game mechanics as well as the main ways in which CRS systems could attack and defend. We refer the interested reader to a presentation by the program manager[1] for a more comprehensive presentation of CGC.

Game Structure

The CGC was structured similarly to a networked "capture-the-flag" competition. Players raced to find, exploit, and fix software bugs in their services in an adversarial environment in real time. The competition started

 Copublished by the IEEE Computer and Reliability Societies

CRS: A hybrid human-machine approach for detecting vulnerabilities

Tuesday, August 20, 2019 | ANNOUNCEMENTS

Galois Awarded $8.6 Million DARPA Contract To Build Cyber Reasoning Tool that Discovers Security Vulnerabilities

Galois will partner with Harvard University and Trail of Bits to build scalable and more cost-effective tools that identify hard-to-find vulnerabilities.

Galois has been awarded an $8.6 million contract by the Defense Advanced Research Projects Agency (DARPA) to build a tool that uses a hybrid human-machine approach to detecting cyber security vulnerabilities that go undetected using traditional methods. The contract was awarded by the DARPA Computers and Humans Exploring Software Security (CHESS) program, which aims to develop capabilities to discover and address vulnerabilities in a scalable, timely, and consistent manner.

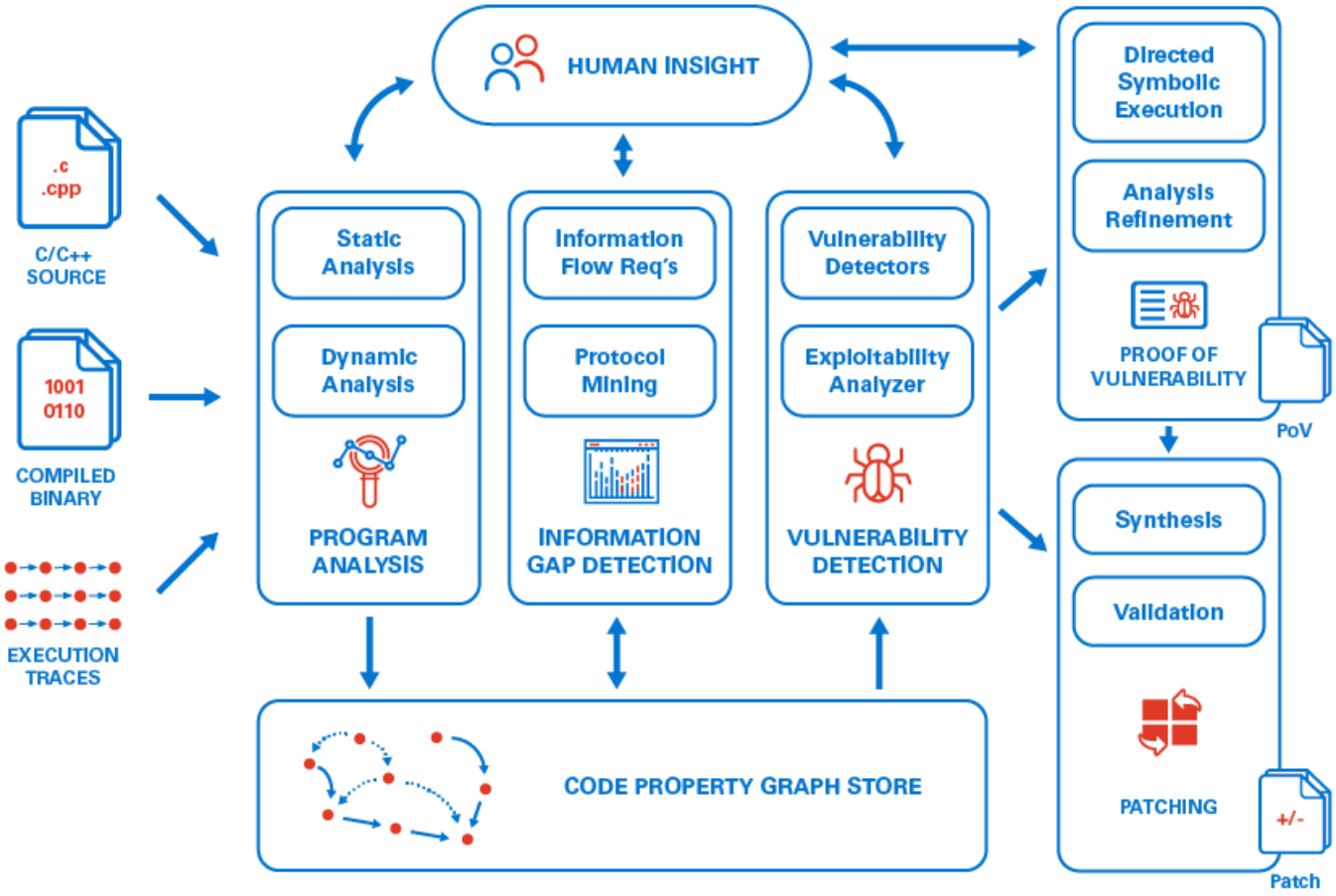
To find and mitigate vulnerabilities in its critical systems, organizations currently rely on security experts who may spend hundreds or thousands of hours reviewing a system to discover a single vulnerability. This process cannot scale sufficiently to secure a continuously growing technology base. Events such as DARPA's 2016 Cyber Grand Challenge have demonstrated the potential for automated security analysis tools to find vulnerabilities without extensive manual effort, but to date these tools have only been capable of detecting a limited set of vulnerability classes and have only worked in limited, controlled settings. By developing a cyber reasoning system that better integrates human insights while retaining the efficiency of automated tools, the CHESS program aims to enable organizations to scale their vulnerability assessment processes to complex, critical systems, such as web browsers and large enterprise applications.

https://galois.com/news/chess-cyber-reasoning-tool-to-discover-security-vulnerabilities/

MATE: Merged Analysis To prevent Exploits

- **MATE** là dự án được Galois phát triển với sự cộng tác của ĐH Harvard và Trail of Bits, nằm trong chương trình "DARPA's Computers and Humans Exploring Software Security (CHESS)".
  - Hướng tới phát triển khả năng **phát hiện và khắc phục lỗ hổng phần mềm** theo phương pháp có thể mở rộng, nhanh chóng và nhất quán.
- Đọc thêm: https://galois.com/project/MATE/

MATE: Merged Analysis To prevent Exploits



- Github: https://galoisinc.github.io/MATE/index.html

Các chủ đề nghiên cứu về Software Security tại InSecLab

- **An toàn phần mềm:**
  - Code Obfuscation
  - Code Similarity Detection
- **Khai thác lỗ hổng phần mềm:**
  - Tự động hóa khai thác lỗ hổng phần mềm
    - Các kỹ thuật Fuzzing
    - Automated Exploit Generation
  - Tự động hóa phát hiện lỗ hổng phần mềm dùng Học máy
    - Fuzzing dùng Mô hình học tăng cường (Reinforcement Learning)/ Học sâu (Deep Learning)
    - Phát hiện lỗ hổng phần mềm bằng Deep Learning kết hợp với Static Code Analysis & Symbolic Execution.
  - Vá lỗ hổng phần mềm tự động

→ Thực hiện *Đề tài NCKH-SV, Đồ án chuyên ngành, Khóa luận tốt nghiệp.*

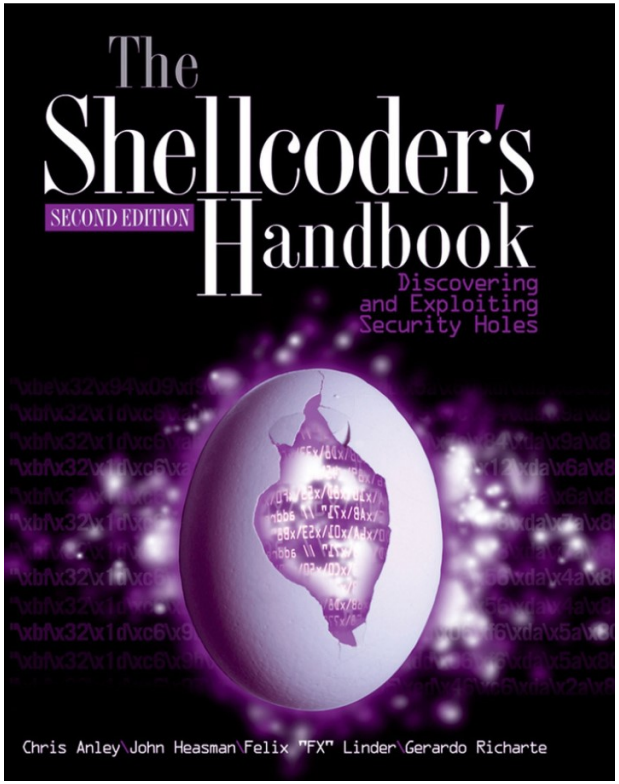
Liên hệ

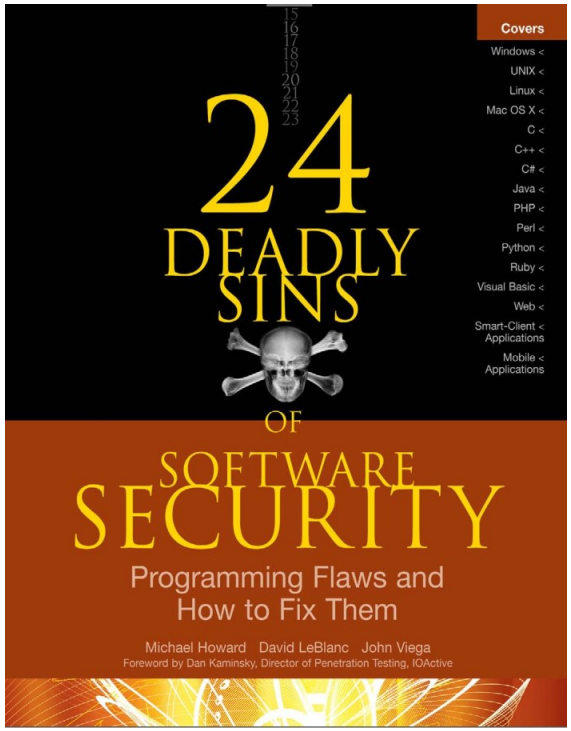
- Email liên hệ:
  - PTN ATTT: inseclab@uit.edu.vn
  - GV: duypt@uit.edu.vn

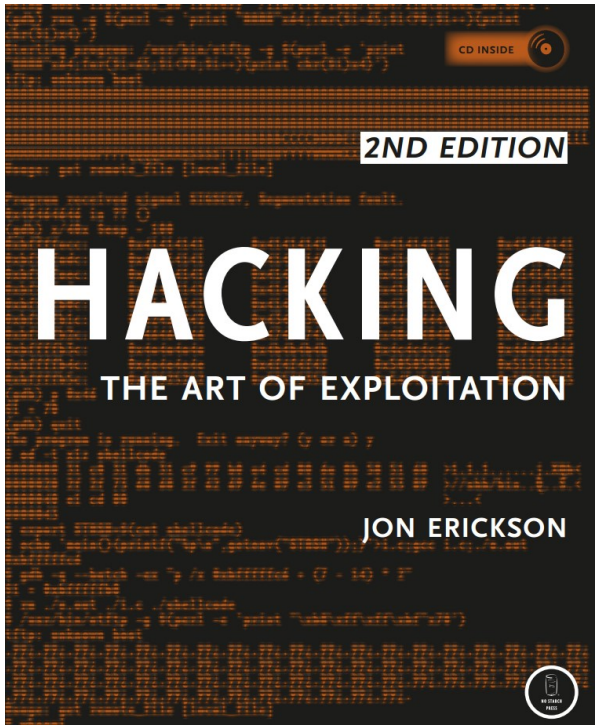
Tài liệu tham khảo: Pwn CTF

- CTF Wiki: https://ctf-wiki.org/pwn/linux/user-mode/environment/
- Modern Binary Exploitation - CSCI 4968: https://github.com/RPISEC/MBE
- NTU Computer Security Fall 2019: https://github.com/yuawn/NTU-Computer-Security
- Nightmare: https://guyinatuxedo.github.io/index.html
- **CS6265:** https://tc.gts3.org/cs6265/tut/tut00-intro.html
- **Educational Heap Exploitation:** https://github.com/shellphish/how2heap
- **Heap Exploitation:** https://techlifecompilation.wordpress.com/2018/10/04/malloc1-heap-exploitation-101/
- **GLibC Heap Exploitation:** https://0x434b.dev/overview-of-glibc-heap-exploitation-techniques/

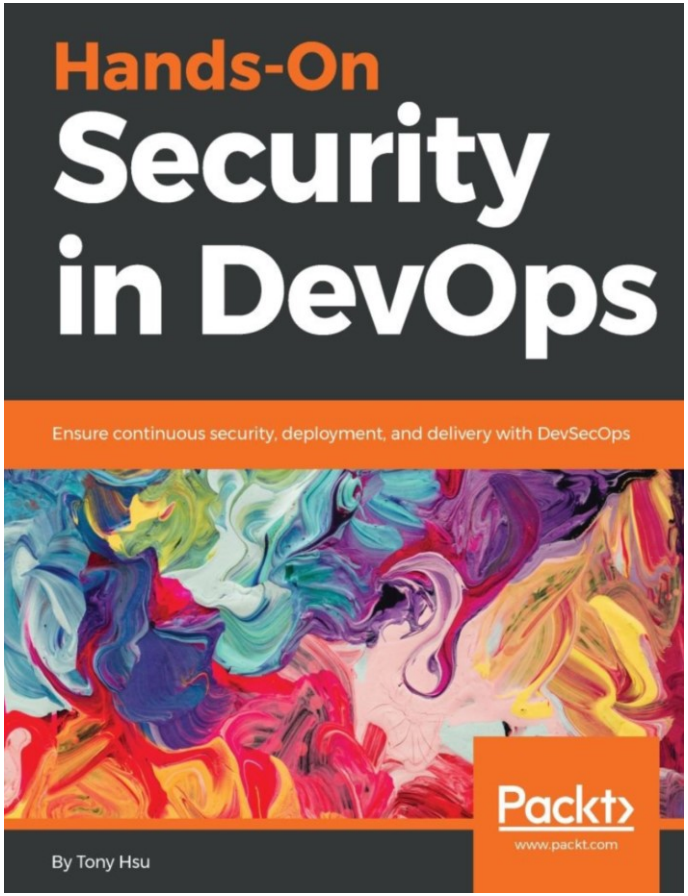
Tài liệu tham khảo

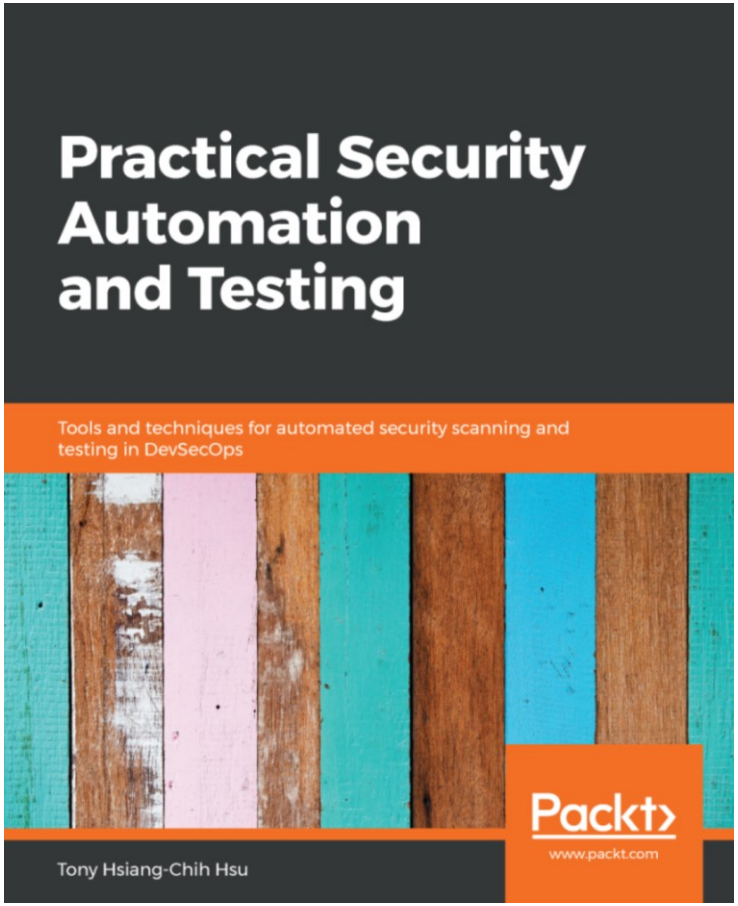




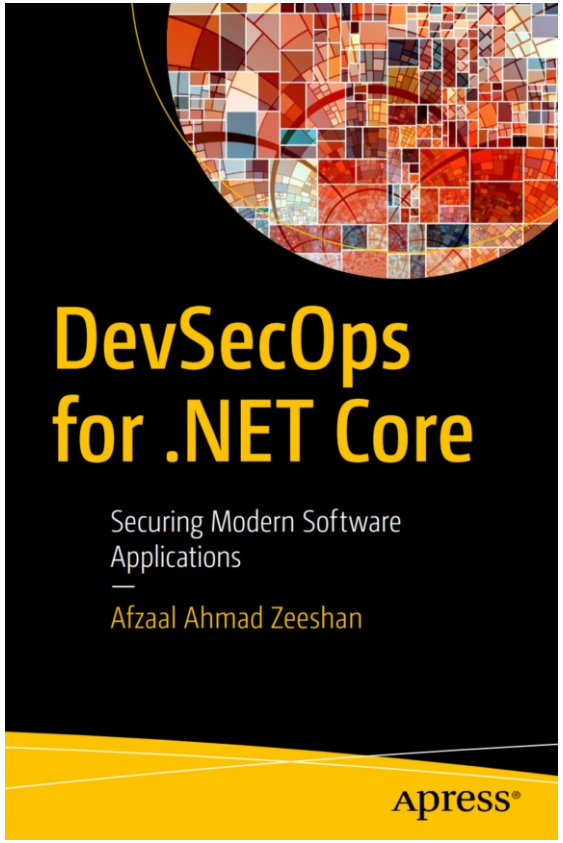


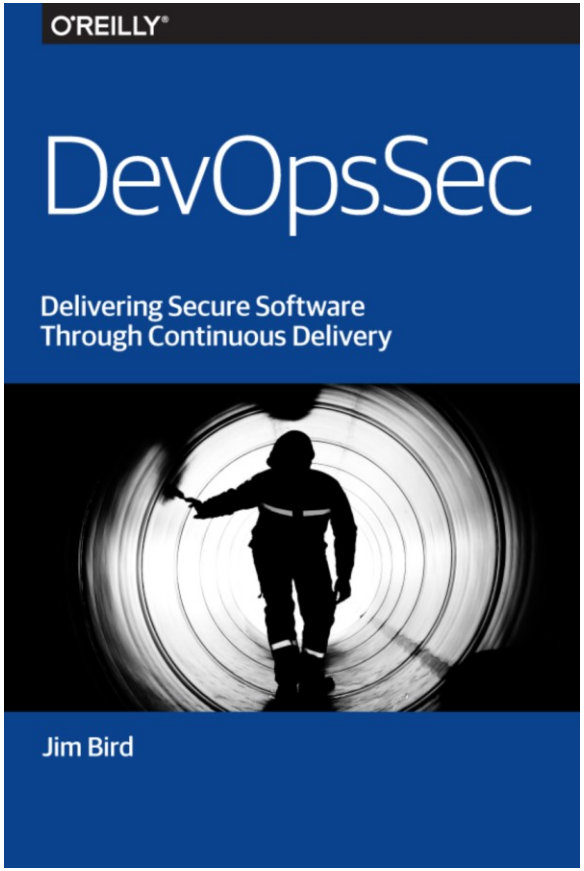
Tài liệu tham khảo



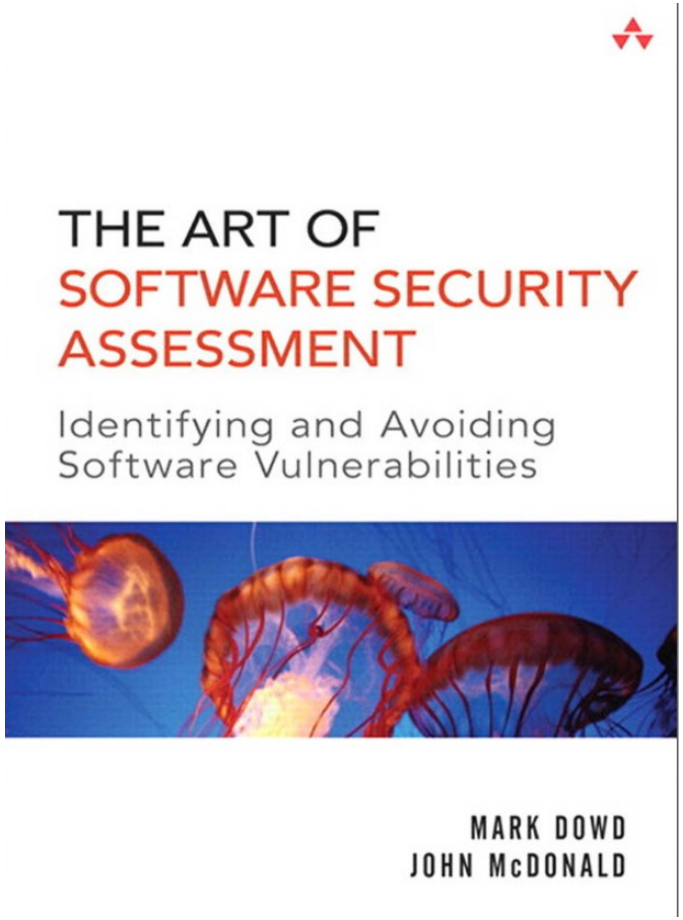


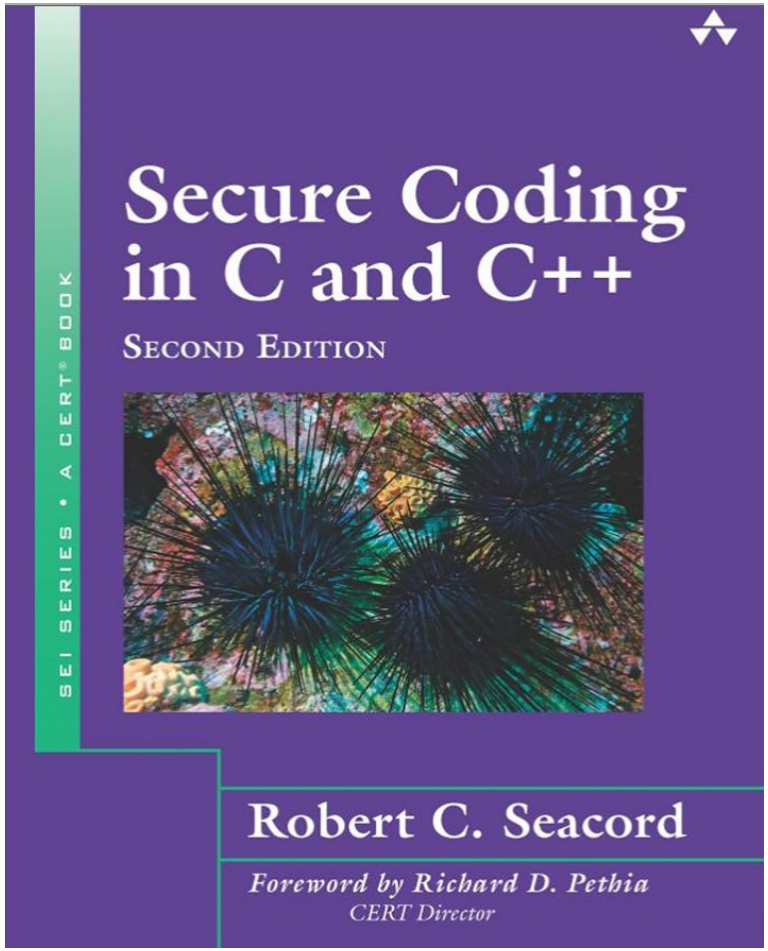
Tài liệu tham khảo



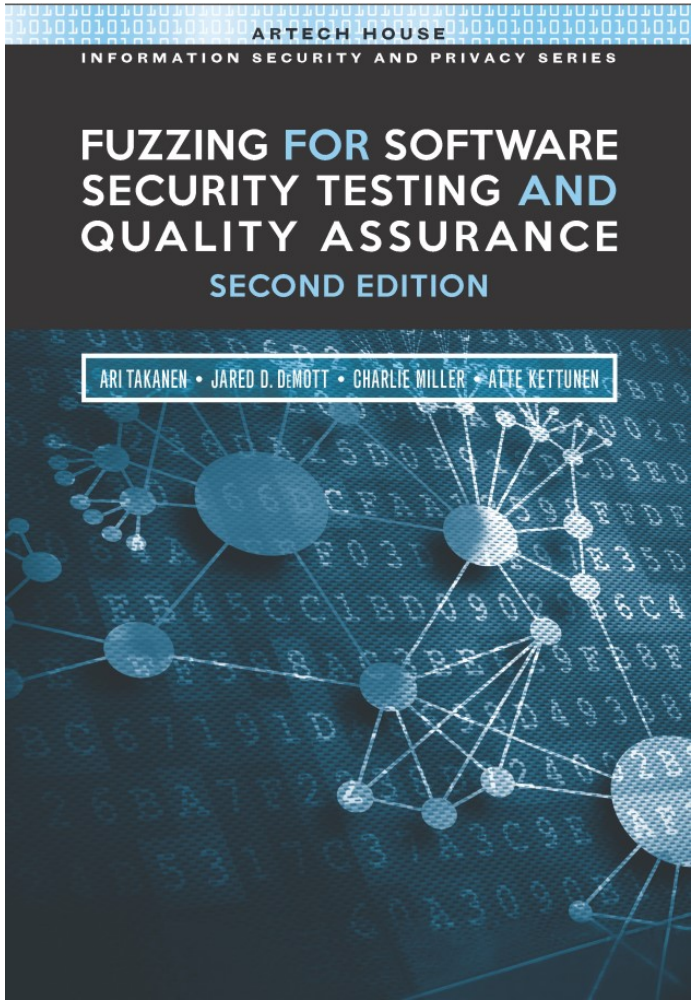


Tài liệu tham khảo

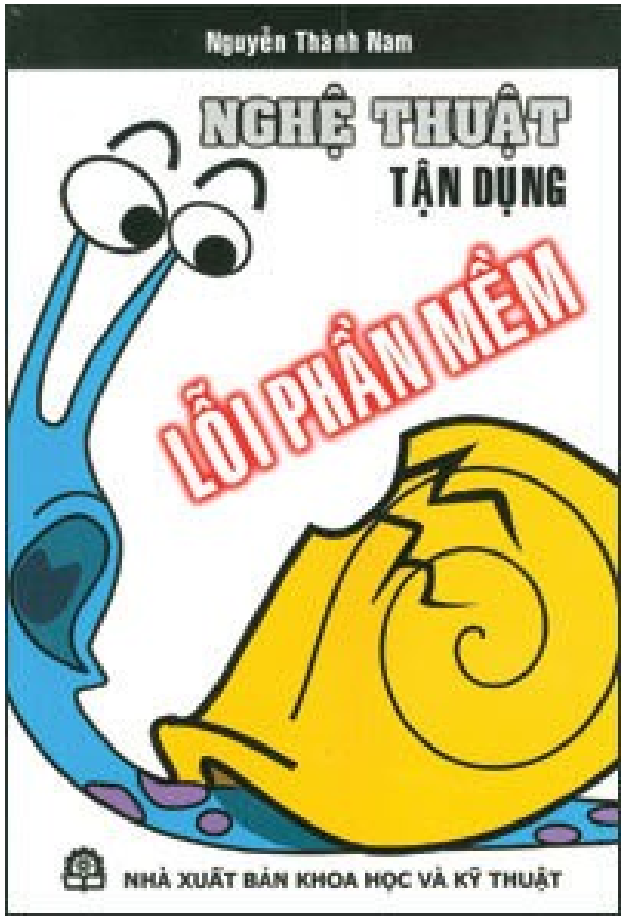




Tài liệu tham khảo



Tài liệu tham khảo



Tài liệu tham khảo

- https://security.berkeley.edu/secure-coding-practice-guidelines
- https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices
- https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf
- https://www.softwaretestinghelp.com/guidelines-for-secure-coding/
- http://security.cs.rpi.edu/courses/binexp-spring2015/
- https://www.ired.team/

Lập trình an toàn & Khai thác lỗ hổng phần mềm

INSECLAB

Trường ĐH CNTT TP. HCM