1 Thema

Entwicklung **a**utomatischer **T**ools zur Lösung von **CTFs**

2 Zeitraum

Sommersemester 2019 und Wintersemester 2019 / 2020

3 Veranstalter

Simon Dierl <simon.dierl@cs.tu-dortmund.de>, Tel. 7543
Malte Mues <malte.mues@tu-dortmund.de>, Tel. 7544
Prof. Dr. Falk Howar <falk.howar@tu-dortmund.de>, Tel. 7945
Informatik Lehrstuhl XIV, Otto-Hahn-Straße 12, Raum 2.008 / 2.009

4 Aufgabe

IT-Sicherheit ist im Moment ein omnipräsentes Thema: Immer größere Teile der Infrastruktur, die wir täglich nutzen, wird durch IT basierte Systeme gemanagt. Dadurch wird es in Zukunft immer wichtiger diese kritischen IT-Systeme vor unbefugten Zugriff und Missbrauch zu schützen.

Um IT-Systeme wirksam verteidigen zu können, muss man jedoch zunächst verstehen, in welcher Art und Weise diese überhaupt manipuliert werden können. Wie Schneier im ersten Kapitel seines Buches „Beyond Fear“ [13] darstellt, ist Sicherheit in erster Linie ein Trade-off zwischen Aufwand zur Manipulation und Aufwand zum Schutz vor Manipulation. Je nachdem, wie ein Anwender die Kosten auf beiden Seiten einschätzt, fühlt er sich bei der Verwendung des Systems sicher oder nicht.

Angehende IT-Sicherheitsexperten können deshalb nur sinnvolle Konzepte zum Schutz vor IT-Angriffen entwerfen, wenn sie in einem ersten Schritt lernen, wie ein Angreifer zu arbeiten und zu denken. Ein in der IT-Sicherheits-Community verbreiteter Ansatz, mit dem diese Fähigkeiten erworben werden, sind Capture the Flag (CTF)-Challenges [2, 5, 6]. In solchen Challenges übernehmen angehende Experten die Rolle eines Angreifers und müssen Sicherheitslücken in speziell präparierten Systemen finden und ausnutzen, um sich Zugang zu geschützen Informationen (der *Flag*) zu verschaffen.

Ziele. Zunehmend werden in der Industrie Werkzeuge aus dem Bereich der formalen Methoden eingesetzt, um Sicherheitslücken automtisiert zu finden (z. B. [3]). Das Ziel der PG ist es, mehrere didaktische Capture the Flag Challenges aufzubauen, die sowohl von manuell als auch automatisiert (mit Hilfe von formalen Methoden) gelöst werden können. Dabei soll der didaktische Fokus des Challenges sowohl auf der Vermitllung der jeweiligen Art von Schwachstelle, als auch auf der Vermittlung der passenden formalen Methode für die automatisierte Lösung liegen.

Vorgehen. Um didaktische Capture the Flag Challenges zu entwerfen, sind drei Schritte notwendig. Zunächst muss man potentielle Schwachstellen identifizieren, dann theoretisch untersuchen, wie man diese entdecken und schließen kann bzw. vermeidet und zuletzt eine wirkungsvolle Abwehr umsetzen.

Sicherheitslücken identifizieren. Bevor man sich mit der Entwicklung von Werkzeugen zur automatisierten Detektion von Sicherheitslücken auseinandersetzen kann, muss man verstehen, welche Klassen von Sicherheitslücken es gibt und wie Angriffe gegen diese funktionieren. Dazu sollen die Teilnehmerinnen und Teilnehmer der PG zunächst verschiedene didaktische CTFs

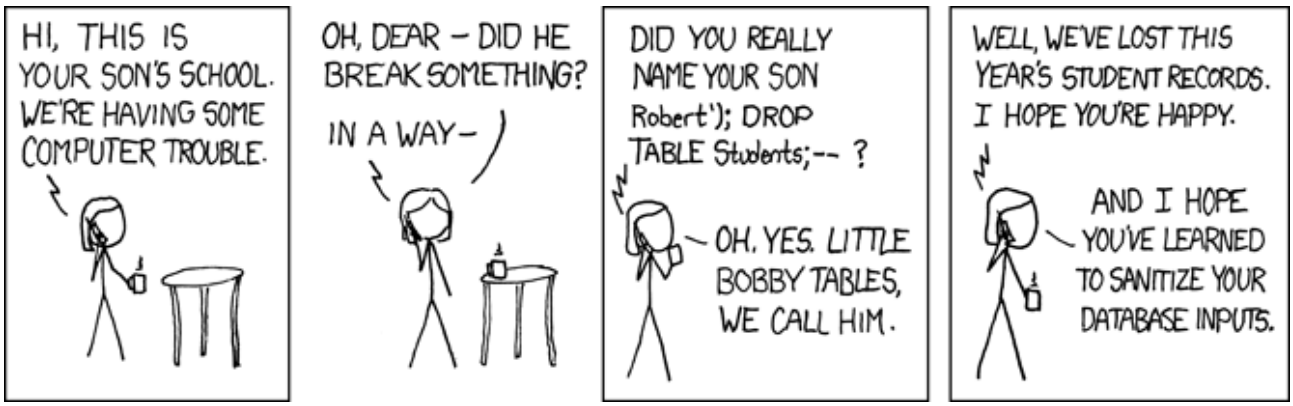

Abbildung 1: Exploits of a Mom[1]

aufarbeiten. Ein Beispiel für eine solche didaktische CTF ist der vom Open Web Application Security Project (OWASP) entwickelte Juice Shop; ein anderes Beispiel ist die didaktische CTF-Challenge der University of Birmingham [2].

Neben dem Wissen, welches in CTF erlebbar gemacht wird, gibt es auch verschiedene best practices, die in der Literatur beschrieben sind. Manche davon, wie die Prüfung von SQL-Eingaben, haben es zwar in das allgemeine Bewusstsein gebracht (vgl. Abbildung 1), sind aber trotz aller humoristischen Betrachtung nach wie vor ein ernstzunehmendes Problem (vgl. OWASP Top 10 List [11]). Weitere große Datenbanken, die zur Bestimmung von Mustern in Sicherheitslücken gesichtet werden sollen, sind das Common Vulnerabilities and Exposures (CVE)-Verzeichnis[2] und das Common Weakness Enumeration (CWE)-Verzeichnis[3].

Methoden zum Entdecken und Schließen von Sicherheitslücken. Nachdem man erkannt hat, was Sicherheitslücken *sind*, muss man diese zuverlässig *finden*, um sie ausnutzen oder vermeiden zu können. Hier gibt es verschiedene Ansätze im Bereich der (semi-)formalen Methoden, die in der PG betrachtet werden sollen. Für Automatenlernen mit der LearnLib [10] wurde z. B. erfolgreich gezeigt, dass diese geeignet ist, um Fehler in Protokollen zu finden [7]. Für Methoden, die auf Automatenlernen basieren, bleibt zu zeigen, dass die gelernten Modelle valide Hypothesen des echten Systemverhaltens sind, um zu voll formalen Methoden zu gelangen. Erste Schritte in diese Richtung wurden in [9] gezeigt.

Byron Cook diskutiert die Effektivität von formalen Methoden zur Detektion von Sicherheitslücken in Softwarekonfigurationen bei der Firma Amazon Web Services (AWS) [3]. Die PG könnte z. B. dem Beispiel von AWS folgen und versuchen, ein formales Model für TomCat- und Datenbankkonfigurationen zu entwerfen. Mit diesem könnten falsch konfigurierte Services identifiziert werden.

Byron Cooks Gruppe bei AWS hat ebenfalls gezeigt, dass Bounded Model Checking geeignet ist, um Sicherheitslücken zu finden bzw. Code zu härten [4]. Ein offenes Problem dabei, welches auch für Symbolic Execution Engines gilt, ist die Modellierung der Umgebung der analysierten Software. Je nach der zu analysierenden Software kann die Umgebung Hardware-Komponenten, Memory Mapper oder andere Bibliotheksfunktionen sein. Z. B. modelliert die symbolische Ausführung in KLEE [1] Aufrufe in die Standardbibliothek mit symbolischen Funktionen.

Die PG soll untersuchen, inwiefern diese Modelle automatisch aus vorhandenen Artefakten erzeugt werden können, um die Verifikation zu beschleunigen und die Anwendung in neuen Um-

[1] `https://xkcd.com/327/`
[2] `https://cve.mitre.org`
[3] `https://cwe.mitre.org`

gebungen zu vereinfachen, z. B. durch den Einsatz von Automatenlernen (s. o.). Es soll ebenfalls untersucht werden, ob Modellierung genutzt werden kann, um Umgebungsmodelle für eines der oben genannten Verfahren zu erstellen. Beispiele dazu wären die Beschreibung von Dateisystemen durch formale Spezifikationen in SibylFS [12] oder die Nutzung von Hybridautomaten [8] zur Beschreibung des Verhaltens von Hardwarekomponenten.

Neben den auf (semi-)formalen Methoden basierenden Werkzeugen sollen auch bestehende Sicherheits- und Penetrationstest-Anwendungen in der PG berücksichtigt werden. Die Studierenden sollen den Umgang mit BlackArch, Kali Linux, Metasploit und anderen klassischen Penetrationstest-Werkzeugen erlernen, die auf Heuristiken oder Brute-Force-Methoden basieren. Zusätzlich sollen vorhandene Source Code-Analyse-Tools wie Valgrind oder Compiler-Plugins in GCC und LLVM genutzt werden.

Das Ziel ist, all diese Werkzeuge zusammen mit formalen Methoden so in den Kontext zu setzen, dass die entstehende Kombination Penetrationstests automatisch ausführt. Sie soll möglichst viele Schwachstellen mit möglichst geringer Beteiligung des Nutzers finden.

Wirkungsvolle Abwehr. Die reine Werkzeugentwicklung macht nur Sinn, wenn die Teilnehmer auch den geschaffenen Mehrwert zeigen können. Eine Möglichkeit zur Evaluation der Leistungsverbesserung ist die SARD Benchmark Suite des National Institute of Standards and Technology[4]. Diese enthält bereits eine Sammlung von verschiedenen bekannten Softwaredefekten und auch eine virtuelle Maschine, in welcher Software mit dokumentierten Sicherheitslücken vorinstalliert wurde. Eine weitere Sammlung von CTF-VMs findet sich auf VulnHub[5].

Die bestehenden CTF Challenges sind meistens dazu gedacht, dass Menschen mit diesen für IT-Sicherheit sensibilisiert und ausgebildet werden sollen. Das Ziel der PG ist jedoch, dass die Studierenden versuchen möglichst viel von dem Prozess zu automatisieren und reproduzierbar zu machen. Dazu sollen die Studierenden aus den gesammelten Sicherheitslücken solche auswählen, die gut von verschiedenen Werkzeugen erkannt werden und daraus eine CTF-Challenge zur Evaluation von automatisierten Penetrationstestwerkzeugen bauen. Anschließend sollen die Studierenden ihre Werkzeugkette an den CTF-Challenges demonstrieren.

5 Teilnahmevoraussetzungen

Folgende fundierten Kenntnisse sind für die Teilnahme an der Projektgruppe *notwendig*:

- Mindestens eine „klassische“ Programmiersprache (z. B. Java, C, C++, C#)
- Mindestens einer Skriptsprache (z. B. Python, bash, PHP)

Folgende Kenntnisse sind für die Teilnahme an der Projektgruppe *hilfreich*, werden jedoch nicht vorausgesetzt:

- **Sicherheitslücken:** Tiefere Kenntnisse über IT-Sicherheitslücken und ihre Vermeidung
- **Formale Methoden:** Grundkenntnisse im Bereich der formalen Methoden wie symbolische Ausführung, Automatenlernen und Codeanalyse.
- **Benchmarking:** Grundkenntnisse im Design von Systembenchmarks.
- **Infrastruktur:** Weiterführende Kenntnisse im Bereich von IT-Infrastruktur wie Netzwerke, Virtualisierung und Betriebssysteme.

Die Grundlagen für hilfreiche Kenntnisse werden im Laufe des Projekts vermittelt.

6 Minimalziel

Die Minimalziele der PG sind:

[4] `https://samate.nist.gov/SARD/testsuite.php` [5] `https://www.vulnhub.com`

- Die PG trifft eine Auswahl an didaktischen Beispielen, anhand derer sie eine automatisierte Werkzeugkette entwickeln wollen.
- 2 VM-Images mit Benchmark-CTFs zur Evaluation von automatisierten IT-Sicherheitsanalyse-Werkzeugen. Jede der VMs muss mindestens 4 verschiedene Schwachstellen enthalten.
- Ein Skript, welches automatisiert die oben beschriebenen zwei VMs analysiert und die dokumentierten Schwachstellen findet.

7 Literatur

[1] C. Cadar, D. Dunbar, D. R. Engler, et al. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In *OSDI*, volume 8, pages 209–224, 2008.

[2] T. Chothia and C. Novakovic. An offline capture the flag-style virtual machine and an assessment of its value for cybersecurity education. *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, 2015.

[3] B. Cook. Formal reasoning about the security of amazon web services. In *International Conference on Computer Aided Verification*, pages 38–47. Springer, 2018.

[4] B. Cook, K. Khazem, D. Kroening, S. Tasiran, M. Tautschnig, and M. R. Tuttle. Model checking boot code from aws data centers. 2018.

[5] C. Cowan, S. Arnold, S. Beattie, C. Wright, and J. Viega. Defcon capture the flag: Defending vulnerable code from intense attack. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 120–129. IEEE, 2003.

[6] C. Eagle and J. L. Clark. Capture-the-flag: Learning computer security under fire. Technical report, Naval Postgraduate School Monterey, CA, 2004.

[7] P. Fiterău-Broştean and F. Howar. Learning-based testing the sliding window behavior of tcp implementations. In *Critical Systems: Formal Methods and Automated Verification*, pages 185–200. Springer, 2017.

[8] T. A. Henzinger. *The Theory of Hybrid Automata*, pages 265–292. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[9] F. Howar, D. Giannakopoulou, M. Mues, and J. A. Navas. Generating component interfaces by integrating static and symbolic analysis, learning, and runtime monitoring. In *International Symposium on Leveraging Applications of Formal Methods*, pages 120–136. Springer, 2018.

[10] M. Isberner, F. Howar, and B. Steffen. The open-source learnlib. In *International Conference on Computer Aided Verification*, pages 487–495. Springer, 2015.

[11] OWASP. Owasp top 10 - 2017: The ten most critical web application security risks. *Sl: The OWASP Foundation*, 2017.

[12] T. Ridge, D. Sheets, T. Tuerk, A. Giugliano, A. Madhavapeddy, and P. Sewell. Sibylfs: Formal specification and oracle-based testing for posix and real-world file systems. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 38–53, New York, NY, USA, 2015. ACM.

[13] B. Schneier. *Beyond fear: Thinking sensibly about security in an uncertain world*. Springer Science & Business Media, 2006.

8 Rechtlicher Hinweis

Die Ergebnisse der Projektarbeit und die dabei erstellte Software sollen der Fakultät für Informatik uneingeschränkt für Lehr- und Forschungszwecke zur freien Verfügung stehen. Darüber hinaus sind keine Einschränkungen der Verwertungsrechte an den Ergebnissen der Projektgruppe und keine Vertraulichkeitsvereinbarungen vorgesehen. Die Teilnehmer der Projektgruppe erkennen jedoch die Dual-Use-Problematik bei der Entwicklung von Penetrationstest-Werkzeugen an und verpflichten sich, die Ergebnisse ausschließlich innerhalb der Grenzen des Strafgesetzbuches zu verwenden.