Hochschule Konstanz
Fakultät Informatik

Software Security

AIN

Hanno Langweg

02 Offensive Security

CAPEC

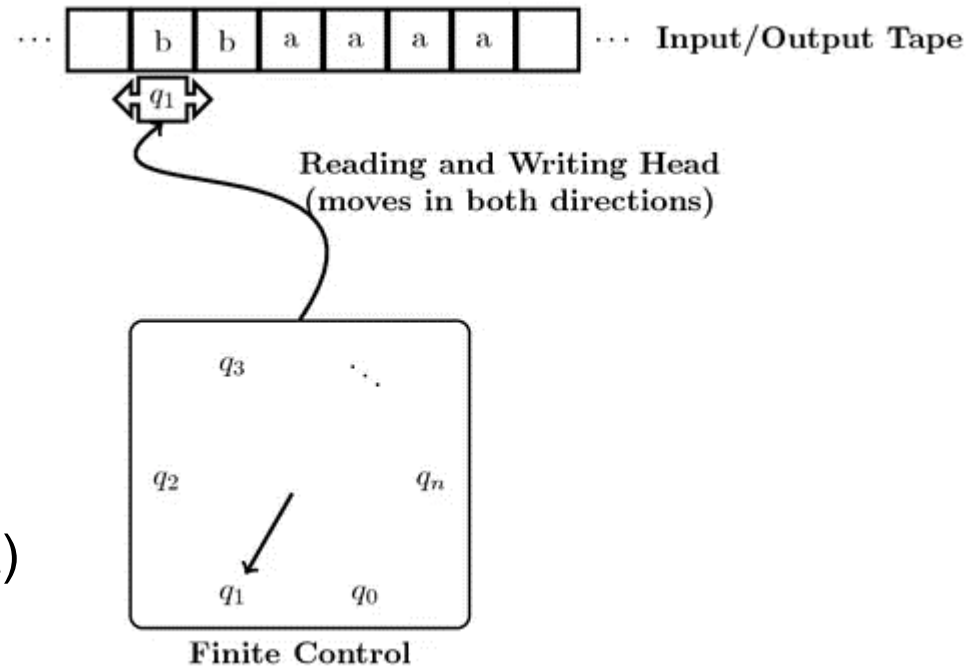
- **C**ommon **A**ttack **P**attern **E**numeration and **C**lassification
- Collection of 500+ attack patterns and techniques
- Related to CWEs (Common Weakness Enumeration)
  ➔ Link attack (CAPEC-ID) to vulnerability (CWE-ID)

- Helps in understanding attacks and attackers' mindset
- Helps in choosing preventive measures
  - Design application
  - Configure environment

CAPEC: Categories of attack mechanisms

- Planning of attacks, collection of information
- Input handling
  - Deceptive interactions, spoofing [authenticity]
  - Injection [integrity]
  - Abuse of existing functionality [availability]
  - Probabilistic techniques: brute force/fuzzing
- Subversion of access control
- Manipulation of data structures (internal state)
- Manipulation of timing and state; race conditions
- Manipulation of system resources (environment)

Attack vectors



- Code (transition function)
- Internal state
- Input
- User (provides code, input)

➔ Identify trust boundaries, interfaces

https://i0.wp.com/tuxar.uk/wp-content/uploads/2017/04/turing-machine-2.png

Input: Fuzzing

- Providing **random input** to program
- **Observing** how input leads to **changes in control flow**

- Most likely results:
  - Program **does not accept input**
  - Program **crashes** (➔ availability; integrity)
  - Program **reveals data** (➔ confidentiality)
  - **Exception** is thrown, security mechanisms are **bypassed**

- Can prove existence of vulnerabilities, but not their absence

Reverse engineering

- Reverse engineering: find out rules of a machine/program by **looking** only at machine/program and its **behaviour** without access to sourcecode
- Find out how **input is processed** and impacts control flow
  - Input formats, filters, checks; whitelisting/blacklisting
- Find out about **resources**, **protocols**, **interfaces** that are used

- More **systematic** than fuzzing, **uncover code paths** that are hard (i.e. unlikely) to trigger by random input
- **Insight** into program behaviour

- Patching: Modify program behaviour
  - Add features, remove vulnerabilities, remove content protection

Reverse engineering: tools

- Decompiler
  - Tool that converts machine code into **source code**
  - Original names of methods, parameters, variables often not retrievable from binary
- Disassembler
  - Tool that converts machine code (machine-readable) into **assembly language** (human-readable)
  - Reveal control flow, call graphs, code patterns
- Debugger
  - Tool that attaches to and **controls processes**
  - Breakpoints, step-wise execution, internal state (memory)
- Virtual machine
  - Similar to program execution by debugger
  - Observe program behaviour during **controlled execution**

Reverse engineering: methods

- Tracing input from interfaces to sinks in program
  ➔ Where is input used in security-relevant decisions?
- Analysis of version differences
  ➔ Compare original and patched version of program
- Code coverage
  ➔ Determine which parts of code are most relevant for further analysis

- Points of concern: Use of APIs, data in shared memory, access to handles, identification of potentially unprotected resources

CTF

- CTF: Capture The Flag
  - Solve puzzles, find/exploit/fix **vulnerabilities**
  - Solution = character sequence ("flag")
- Traditional **categories**: web, reversing, crypto, forensics
  100-500 points based on estimated difficulty
- Single evening or up to 48 hours; **often online**, sometimes on-site
- Styles
  - **Jeopardy**: central server, no interaction between competing teams
  - **Attack/Defence**: Teams attack each other, defend own server/services
- **Gamified training** in vulnerability detection/exploiting; attacks' mindset

- Upcoming competitions: https://ctftime.org