Vulnerability assessment

Tools & methodology

WIP report #1

Chariton Karamitas
huku@census-labs.com
https://github.com/huku-

Dafuq?

- What?
  - Choose target software and evaluate its ability to handle specially crafted inputs
- How?
  - Read available documentation and try to understand how it works
  - Fuzz the hell out of it
  - Read or reverse its code, locate input processing points and look for vulnerabilities (manual way)
  - Apply formal methods and try to automatically find vulnerabilities (automated way)

Doin’ it like a pro

- Doing it for hobby vs. doing it professionally
- Hobby?
  - Personal time management
  - Look for vulnerabilities in whatever you want to own
- Professionally?
  - Strict deadlines; customer expects certain (impressive and persuading) results
  - Be the customer’s little bitch and look for vulnerabilities in whatever he wants to own :-P
  - ...maybe he wants to own **you** :)

Problem?

- It takes time
- Time = $$.$$$:)
- We need...
    - ...a set of methodologies to follow
    - ...a set of tools to use

Methodology I

- Setup the target software in a fully updated working environment (hardware, OS, software dependencies, configurations)
- Start using it, observe **how** it works and try to **understand why** it works that way
    - Develop scenarios and use cases
- Determine input formats and collect input samples
    - Crawl the web
    - Manually construct test cases

Methodology II

- Fuzz the target using the collected test cases (plenty of tools & techniques)
- Use tools targeted at manual analysis **to understand software internals and design choices made by the developers**
  - Got source? **OpenGrok** or any other cross referencing system
  - Got binary? **IDA Pro**, **metasm**, whatever...
  - Locate input sources and obscure features
  - Try to think what the developer may have done wrong and make some $.$$$, $$.$$$ or even $$$.$$$:)
  - Use tools' scripting capabilities to automate simple tasks

Methodology III

- Convert binary & source code to IR suitable for automated analysis
  - Model the program using some theory
  - Try to prove the existence of certain properties in the program
    - Vulnerability classes can be modeled
  - Don’t try to design a fully automated system
    - **Build software that will help you during manual analysis**

Tools

- Step #1 - Pick a correct & useful disassembler
  - Endless choices (most of them incomplete)
  - Best choices: **XED2** (Intel) & **capstone** (multiple architectures)
  - Focus on x86 and AMD64
  - Imo this is the most important step
- Step #2 - **pyxed**
  - Python bindings for Intel's XED2
  - https://github.com/huku-/pyxed

Tools

- Hm... **pyxed**? What's next?
  - CFG (almost done) & HTML5 based interactive graphs (easy)
  - Dominator trees
  - Loop analysis
  - Program slicing
  - Taint analysis
  - Binary to IR translator
    - REIL, ELIR, ESIL, LLVM, VEX?

Questions?