Fuzzing Attack

What is Fuzzing ?

- Fuzzing is a **black-box** software testing technique and consists of finding implementation flaws and bugs by using malformed/semi-malformed payloads via automation
- Fuzzing is the art of **automatic bug finding**, and it's role is to find software implementation faults, and identify them

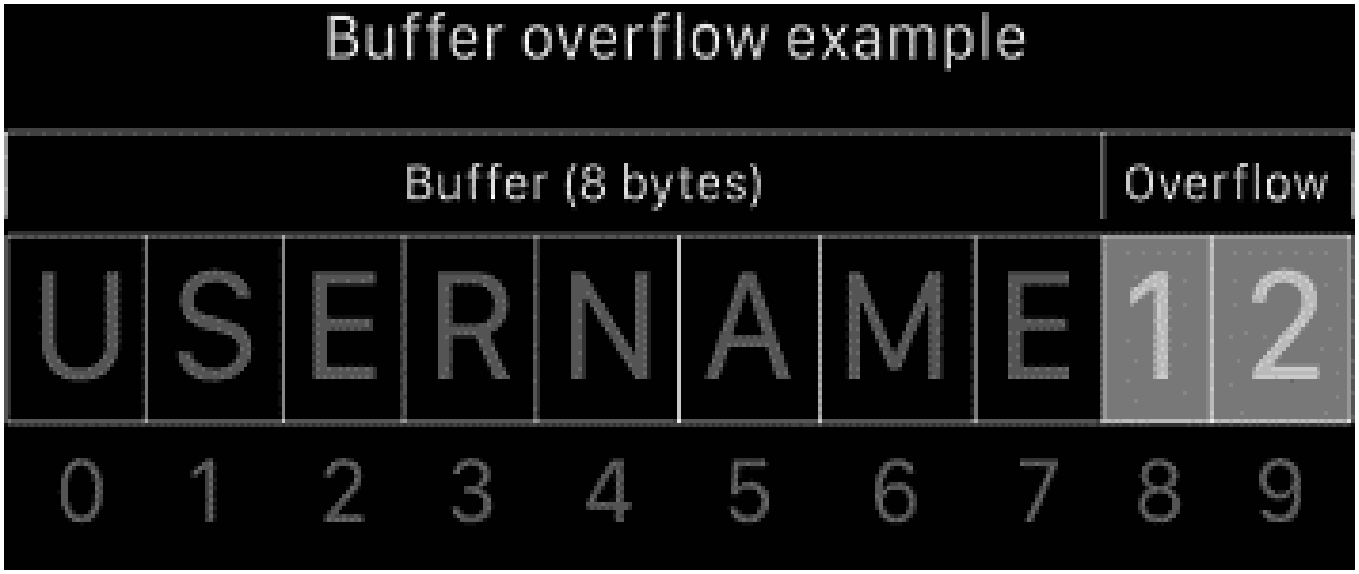
Fuzzing

- Fuzzing is an aging mechanism developed at the University of Wisconsin – Madison in 1989 by Professor Barton Miller and his students.
- Fuzzing is a means of detecting potential implementation weaknesses that can be used to take advantage of any target.
- To do this, a specific fuzzer must be used, where semi-random data is injected into a program/stack to detect bugs or crashes

Fuzzing : Example

- Let's take **an FTP application as an example**. If the size of the username string is equal to eight bytes, the max size of a string can be: **infosec1** (eight characters) or **username**,
- When an automation program (the fuzzer) sends arbitrary payloads during the authentication process, several username payloads can be generated with different sizes and character sets, or even templates.
- If the username parameter is greater than eight bytes, the application will crash, and that will create a buffer overflow condition.
- From here, a remote code execution vulnerability could be explored via the execution of a crafted payload.

Buffer overflow vulnerability:Fuzzing



Fuzzing workflow

Fuzzing Workflow

- Step 1:Initially, the target system must be identified to select the specific fuzzer, the target input, and the right character-set inputs in order to generate the final payloads to test.
- Step 2: After the target input identification, the payload list is generated. Several types of data can be included, such as strings, digits, characters and combinations between them within different input sizes.
- Step 3: Next, the payloads are executed by starting the fuzzer in the right conditions.

Fuzzing Workflow

- Step 4: A crucial part of this process is monitoring the system behaviours and log defects. Here is where we analyze — via an offline format — the results of the test. The potential flaw, bug or crash is detected in this phase.
- Step 5 : Finally, the specially crafted payload can be created according to the results obtained from the fuzzing process to take advantage of the target system, resulting in the creation of the final exploit.

Fuzzing Types

- Application Fuzzing
- Protocol Fuzzing
- File Format Fuzzing

Application Fuzzing

Every input can be fuzzed (inputs, URLs parameters, forms, cookies and so on) with different character sets and payloads but the same goal: crash the system to take advantage of the implementation weakness

Protocol Fuzzing

- A protocol fuzzer sends forged packets to the tested application, or eventually acts as a proxy, modifying requests on the fly and replaying them (e.g., Burp Suite tool — proxy feature).

File Format Fuzzing

- A file format fuzzer can generate multiple malformed samples and opens them sequentially. After that, the program crashes and the debug information is kept for further investigation.
- This kind of fuzzer is less common but still tends to appear these days. For example, MS04-028 (KB833987), Microsoft's JPEG GDI+ vulnerability, is one example of this type of fuzzing scenario.

Fuzzing Tools

- SPIKE
- American Fuzzy Lop
- Radamsa
- Boofuzz
- Bfuzz
- Powerfuzzer