Exploiting software vulnerabilities

Cees-Bart Breunesse

Software vulnerability?

• Unexpected behavior (bug) of a software product given some input:
  – Missing or incorrect input validation.
• Exploitable: unexpected behavior → vulnerability
• An exploit uses a vulnerability to change the path of execution.

Scope

• Local vulnerability:
  – The vulnerability exists on the same machine you’re working on.
  – SUID binaries
• Remote vulnerability:
  – The vulnerability exists on a remote machine.
  – Server apps (web/mail servers)

Local vulnerability

• Main focus: SUID binaries.
• Secondary: kernel vulnerabilities.

• Context:
  – SUID binaries are usually written in C, C++
  – Functions in C can be directly translated to operations on the stack.

Stack example

#include <string.h>
int m(char *s) {
  char buf[10];
  strcpy(buf, s);
  return 0;
}
int main(int argc, char **argv) {
  if(argc > 1)
    m(argv[1]);
  return 0;
}

(C) source code vulnerabilities

• Incorrect bounds checking errors
• Format strings
• Off-by-one errors
• Loops
• Sign errors
Off-by-one error

- OpenBSD ftp daemon

```c
char npath[MAXPATHLEN];
int i;
for(i=0; *name = '\0' && i < sizeof(npath) - 1; i++, name++) {
    npath[i] = *name;
    if (*name == '"')
        npath[++i] = '"';
} npath[i] = '\0';
```

Sign error

```c
#include <string.h>
#define MAXSIZE 255
static char[MAXSIZE] src;
void m(int i) {
    char buf[MAXSIZE];
    if (i < MAXSIZE)
        memcpy(buf, in, i);
    return;
}
void main(int argc, char **argv) {
    m(100); m(300); m(-10);
}
```

Format string bug

- Most likely to be found in logging and debugging code.
- Vulnerable: family of printf/sprintf/… functions.
- `printf(fmt, arg1, arg2, …);`
- `fmt` may contain conversions to print the arguments.
- Special conversion `%n` allows writing the number of currently written bytes to an argument of printf.
- Note that arguments to a function are just stack positions.
- Consequences:
  - Reading out memory, disclosing information
  - Possible change in execution path by change of return address

Finding local vulnerabilities

- `find / -perm +4000`
- Look for:
  - hard to use features
  - almost never necessary features
  - uses of getenv(), temporary files
  - parsings of complex, user controlled data

Remote vulnerabilities

- Main focus: all types of remote services
  - Web servers
    - Web applications
      - Bulletin board
      - Blogs
      - Online agenda
  - Databases
  - Remote services are typically more layered than local services.
    - Php
    - perl
    - dotNET
    - SSL components
  - Input validation (or the lack thereof) takes place at lots of layers in the service.

Techniques to identify vulnerabilities

- Static checking
  - Pro: find many potential problems, but ..
  - Con:
    - finds many false positives (non-exploitable bugs)
    - Source code required
- Formal verification
  - Pro: proves the absence of bugs, but ..
  - Con:
    - Very labor intensive
    - Source code required
- Fuzzing/Fault injection
  - Pro: No source code required
  - Con:
    - Computation time
    - Does not handle “undocumented features”
Fuzzing example

GET http://www.win.tue.nl/ipa HTTP/1.0
Identify the separators or delimiters, and inject a fault right before and right after.

GET<FAULT> http://www.win.tue.nl/ipa HTTP/1.0
GET http://www.win.tue.nl/ipa HTTP/1.0
GET http://<FAULT>/www.win.tue.nl/ipa HTTP/1.0
GET http://<FAULT>/www.win.tue.nl/ipa HTTP/1.0
GET http://www.win.tue.nl/ipa HTTP/1.0

Smart fuzzers use faults that mirror valid data to get past the initial sanity checks.
Even smarter fuzzers use open-source parsers to generate input patterns.
FREE fuzzer: SPIKE (http://www.immunitysec.com)

Past projects

- TIS (Tentamen Informatie Systeem)
- RIES (Rijnland Internet Election System)
- KOA (Kiezen op Afstand)

Current projects

- Infostroom
- CompuWall

CompuWall

- Firewall/gateway proxy to protect LANs from the evil internet.
- Evil internet = Java/JavaScript/ActiveX/archives/M$ documents/…
- Actions performed by CompuWall (on app level):
  - Parse, interpret, and possible change HTML pages.
  - Authentication by means of SSL, IP or MAC.
- System is black box; no source.
- Only possible approach: fuzzing/fault injection.

Infostroom

- Large electricity provider wants to install smart meters that can be read out over de power grid itself.
- Issues:
  - Traffic must be encrypted.
  - Meters must authenticate.
  - Software on the meter must be robust.
  - Meters have 20 year lifespan.
  - One meter may be cracked, but this may not endanger the other meters.