Revisiting Topics

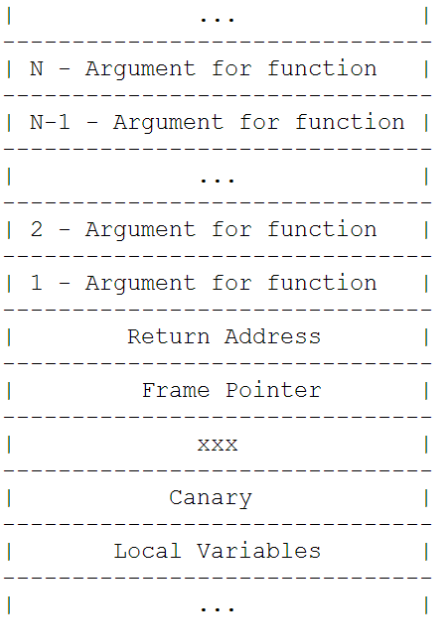
CIS 4930 / 5930. Offensive Computer Security.
Spring 2014

Outline

- Stack cookies
- SSL bugs this year
  - iOS GOTO FAIL
  - GNU Utils
  - OPENSSL…
- Akamai's private key leaked
- SSL, Certs, CA's and the rise of signed malware
- Big picture

Stack Cookies

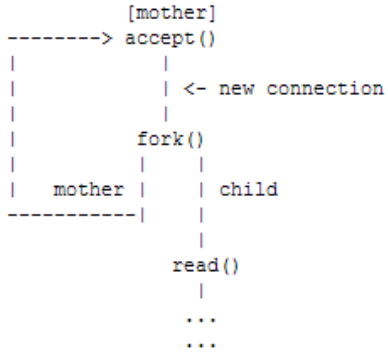
http://phrack.org/issues/67/13.html#article



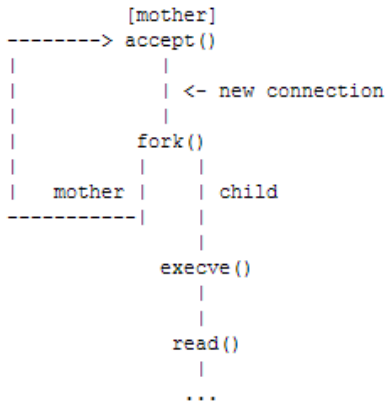
---[3.3 - Exploiting canaries remotely

Usually networks daemons create a new thread by calling clone() or a new process by calling fork() to support a new connection. In the case of fork() and depending on the daemon, the child process may or may not call execve() which means that it will be in one the two situations:

1. without execve()



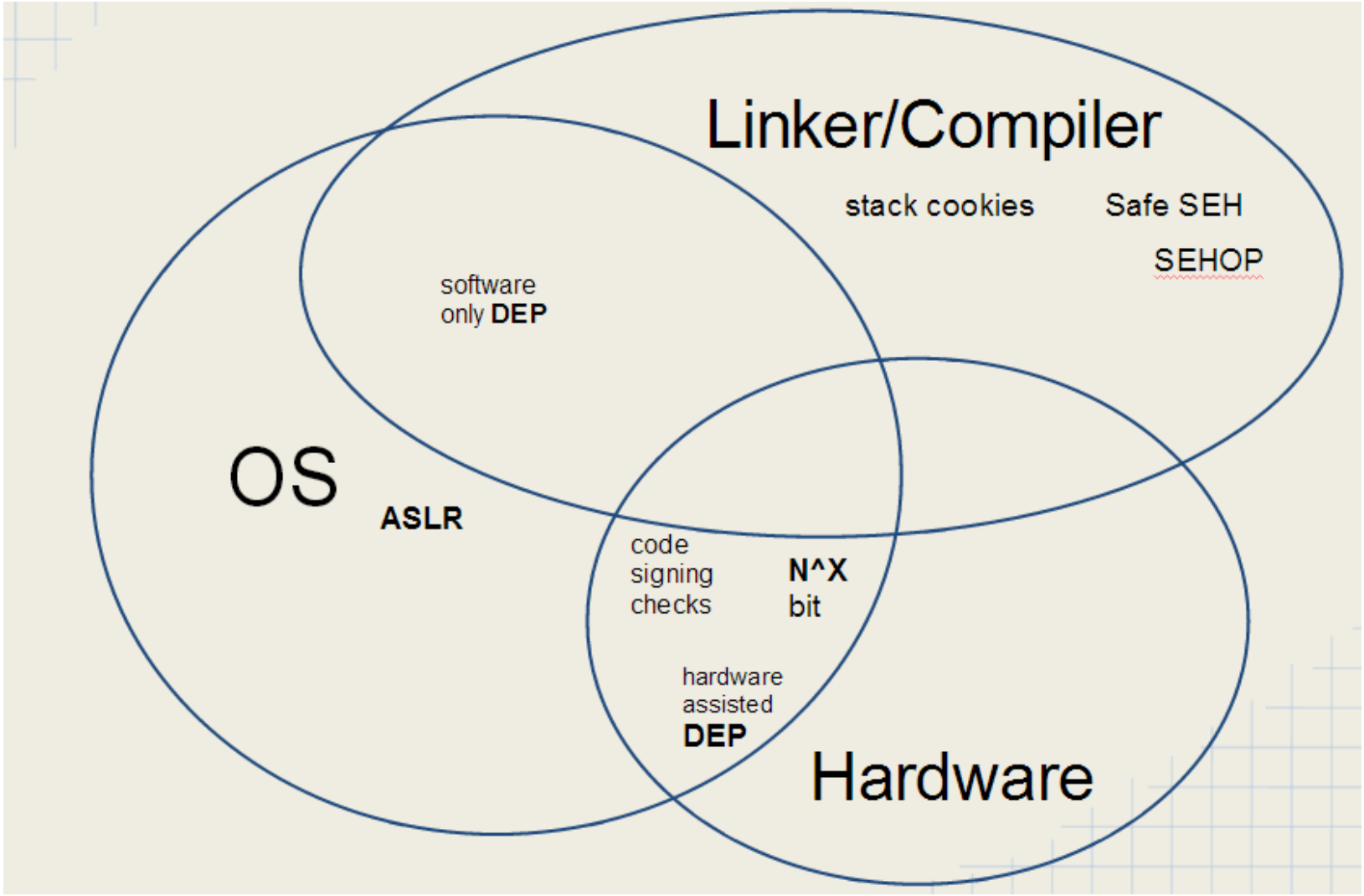
2. with execve()



As stated by the man page:

*) the fork() system call creates a duplicate of the calling process which means that father and child share a same canary as this is a per-process-canary and not a per-function-canary mechanism. This is an interesting property as if for each attempt we were able to guess a little of the canary then with a finite number of guesses we would be successful.

*) when execve() is called "text, data, bss, and stack of the calling process are overwritten by that of the program loaded." This implies that the canary is different for each child. As a result, being able to guess a little of the child canary is most likely useless as this will result in a crash and any result wouldn't be applicable to the next child.


Linker/Compiler
stack cookies
Safe SEH
SEHOP
software
only DEP
OS
ASLR
code
signing
checks
N^X
bit
hardware
assisted
DEP
Hardware

What you should know

```
void foo(...)
{
stack_buffer_overflow()
}
```

How to exploit this:

1. with stack cookies
2. + DEP
3. + ASLR

Require another bug/weakness?

4. Then think about how to further mitigate

What you should know

```
void foo(...)
{
heap_buffer_overflow()
}
```

How to exploit this:

(depends on heap algorithms)

1. with stack cookies
2. + DEP
3. + ASLR

Require another bug/weakness?

4. Then think about how to further mitigate

What you should know

```
void foo(...)
{
integer_bug_+_arbitrary_write()
x[i] = blah
//x[-59] = new return address…
}
```

How to exploit this:

1. with stack cookies
2. + DEP
3. + ASLR

Require another bug/weakness?

4. Then think about other mitigations:
   a. CFI
   b. EMET suite

What you should know

```
void foo(...)
{
// you only control upper half of an address
partial_arbitrary_byte()
}
```

How to exploit this:

1. with stack cookies
2. + DEP
3. + ASLR

Require another bug/weakness?

4. Then think about other mitigations:
   a. CFI
   b. EMET suite

What you should know

```
void foo(...)
{
// you only control upper half of an address
use_after_free()
}
```

How to exploit this:

1. with stack cookies
2. + DEP
3. + ASLR

Require another bug/weakness?

4. Then think about other mitigations:
   a. CFI
   b. EMET suite

What you should know

```
void foo(...)
{
// you only control upper half of an address
use_of_uninitialized_variables()
}
```

How to exploit this:

1. with stack cookies
2. + DEP
3. + ASLR

Require another bug/weakness?

4. Then think about other mitigations:
   a. CFI
   b. EMET suite

What you should know

```
void foo(...)
{
// you only control upper half of an address
format_string_bugs()
}
```

How to exploit this:

1. with stack cookies
2. + DEP
3. + ASLR

Require another bug/weakness?

4. Then think about other mitigations:
   a. CFI
   b. EMET suite

Going Further

Now imagine each of these:

- Local priv esc, inside sandbox, against AV
  - What beats Signature based AV?
- Remote exploitation
  - Through firewall? Through WAF?
    - How does having to work with encoding affect exploitability for each?
      - Does it actually affect exploitability?
      - or just the payload execution/delivery???

SSL Bugs

https://www.imperialviolet.org/2014/02/22/applebug.html

http://arstechnica.com/security/2014/03/critical-crypto-bug-leaves-linux-hundreds-of-apps-open-to-eavesdropping/

http://heartbleed.com/

How Big?

http://mashable.com/2014/04/09/heartbleed-bug-websites-affected/

How Heartbleed Works

http://xkcd.com/1354/

Akamai hit by Heartbleed

http://en.wikipedia.org/wiki/Akamai_Technologies

Had a custom patch for heartbleed; still vulnerable:

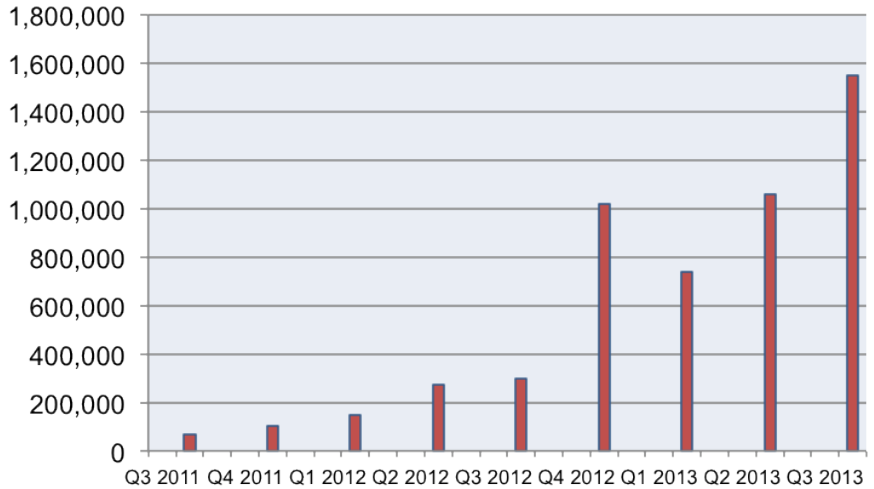
http://www.computerworld.com/s/article/9247650/Akamai_admits_issuing_faulty_OpenSSL_patch_reissues_keys

- Attacked.
- Impact: over 30% of all internet traffic exposed.

Digitally Signed Malware

over 1.5 digitally signed NEW malware in 2013:

https://blogs.mcafee.com/mcafee-labs/digitally-signed-malware-just-what-can-you-trust-now



Big Picture

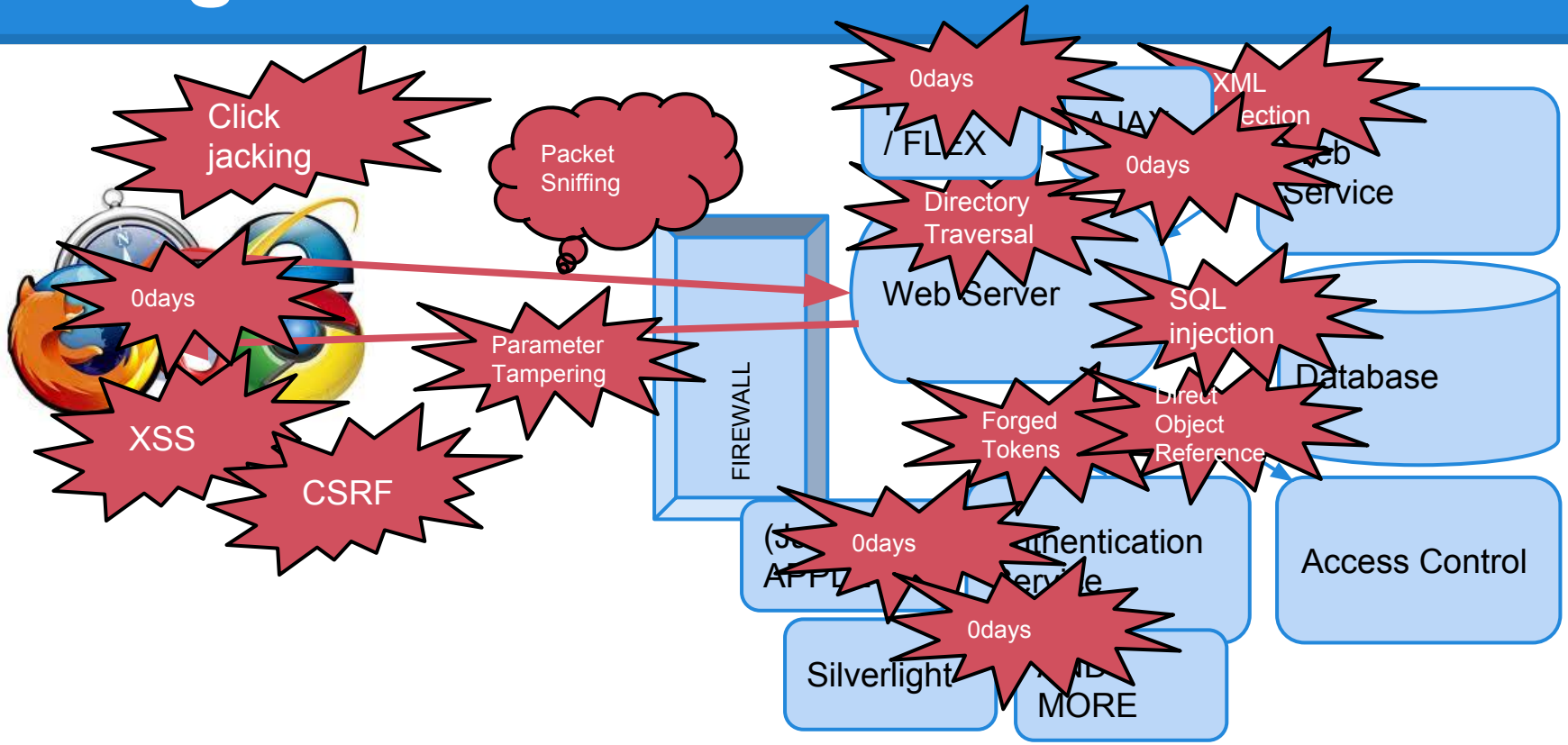
Security

- Confidentiality
- Integrity
- Availability

Exploitation / Hacking grants attackers Access

- DoS to replay attacks to remote code execution

Big Picture


Click jacking
Packet Sniffing
0days
XML ection
/ FLEX
0days
eb Service
Directory Traversal
0days
Web Server
SQL injection
Database
Parameter Tampering
FIREWALL
XSS
CSRF
Forged Tokens
Direct Object Reference
0days
hentication
Access Control
0days
Silverlight
MORE

Next Week

Social Engineering

Physical Security

Take Home Final Exam

Questions?