CSci 4271W
Development of Secure Software Systems
Day 28: Final bonus topics

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

Control-flow integrity (CFI)

Logistics intermission

More modern exploit techniques

More causes of crypto failure

DNSSEC

Some philosophy

- Remember allowlist vs. denylist?
- Rather than specific attacks, tighten behavior
  - Compare: type system; garbage collector vs. use-after-free
- CFI: apply to control-flow attacks

Basic CFI principle

- Each indirect jump should only go to a programmer-intended (or compiler-intended) target
- I.e., enforce call graph
- Often: identify disjoint target sets

Approximating the call graph

- One set: all legal indirect targets
- Two sets: indirect calls and return points
- n sets: needs possibly-difficult points-to analysis

Target checking: classic

- Identifier is a unique 32-bit value
- Can embed in effectively-nop instruction
- Check value at target before jump
- Optionally add shadow stack

Target checking: classic

```
cmp [ecx], 12345678h
jne error_label
lea ecx, [ecx+4]
jmp ecx
```

Challenge 1: performance

- In CCS'05 paper: 16% avg., 45% max.
  - Widely varying by program
  - Probably too much for on-by-default
- Improved in later research
  - Common alternative: use tables of legal targets

Challenge 2: compatibility

- Compilation information required
- Must transform entire program together
- Can't inter-operate with untransformed code

More recent advances: COTS

- Commercial off-the-shelf binaries
- CCFIR (Berkeley+PKU, Oakland'13): Windows
- CFI for COTS Binaries (Stony Brook, USENIX'13): Linux

COTS techniques

- CCFIR: use Windows ASLR information to find targets
- Linux paper: keep copy of original binary, build translation table

Control-Flow Guard

- CFI-style defense now in latest Windows systems
- Compiler generates tables of legal targets
- At runtime, table managed by kernel, read-only to user-space

Coarse-grained counter-attack

- "Out of Control" paper, Oakland'14
- Limit to gadgets allowed by coarse policy
  - Indirect call to function entry
  - Return to point after call site ("call-preceded")
- Use existing direct calls to `VirtualProtect`
- Also used against kBouncer

Control-flow bending counter-attack

- Control-flow attacks that still respect the CFG
- Especially easy without a shadow stack
- Printf-oriented programming generalizes format-string attacks

Outline

Control-flow integrity (CFI)

Logistics intermission

More modern exploit techniques

More causes of crypto failure

DNSSEC

SRT reminder

- Thanks to the approx. half of you who filled out the online SRT
- Others, please consider devoting a bit of time tonight or tomorrow
- `https://srt.umn.edu/blue`

Final project submission

- Two components: fixing patch and revised report
- Take advantage of sample attacks posted on Piazza
- Page limit increased to 6 pages
- Due on Canvas by Wednesday night
  - Left-over extension can extend to Friday night, or on-time for 5% extra credit

Last lab section tomorrow

- There will be a lab section at the normal time tomorrow
- Last scheduled Zoom event of the semester
- Topic: counter-attack against $W \oplus X$ and ASLR

Outline

Control-flow integrity (CFI)

Logistics intermission

More modern exploit techniques

More causes of crypto failure

DNSSEC

Target #1: web browsers

- Widely used on desktop and mobile platforms
- Easily exposed to malicious code
- JavaScript is useful for constructing fancy attacks

Heap spraying

- How to take advantage of uncontrolled jump?
- Maximize proportion of memory that is a target
- Generalize NOP sled idea, using benign allocator
- Under W⊕X, can't be code directly

JIT spraying

- Can we use a JIT compiler to make our sleds?
- Exploit unaligned execution:
  - Benign but weird high-level code (bitwise ops. with constants)
  - Benign but predictable JITted code
  - Becomes sled + exploit when entered unaligned

JIT spray example

```
25 90 90 90 3c   and $0x3c909090,%eax
25 90 90 90 3c   and $0x3c909090,%eax
25 90 90 90 3c   and $0x3c909090,%eax
25 90 90 90 3c   and $0x3c909090,%eax
```

JIT spray example

```
90              nop
90              nop
90              nop
3c 25           cmp $0x25,%al
90              nop
90              nop
90              nop
3c 25           cmp $0x25,%al
```

Use-after-free

- Low-level memory error of choice in web browsers
- Not as easily audited as buffer overflows
- Can lurk in attacker-controlled corner cases
- JavaScript and Document Object Model (DOM)

Sandboxes and escape

- Chrome NaCl: run untrusted native code with SFI
  - Extra instruction-level checks somewhat like CFI
- Each web page rendered in own, less-trusted process
- But not easy to make sandboxes secure
  - While allowing functionality

Chained bugs in Pwnium 1

- Google-run contest for complete Chrome exploits
  - First edition in spring 2012
- Winner 1: 6 vulnerabilities
- Winner 2: 14 bugs and "missed hardening opportunities"
- Each got $60k, bugs promptly fixed

Outline

Control-flow integrity (CFI)

Logistics intermission

More modern exploit techniques

More causes of crypto failure

DNSSEC

Side-channel attacks

- Timing analysis:
  - Number of 1 bits in modular exponentiation
  - Unpadding, MAC checking, error handling
  - Probe cache state of AES table entries
- Power analysis
  - Especially useful against smartcards
- Fault injection
- Data non-erasure
  - Hard disks, "cold boot" on RAM

WEP "privacy"

- First WiFi encryption standard: Wired Equivalent Privacy (WEP)
- F&S: designed by a committee that contained no cryptographers
- Problem 1: note "privacy": what about integrity?
  - Nope: stream cipher + CRC = easy bit flipping

WEP shared key

- Single key known by all parties on network
- Easy to compromise
- Hard to change
- Also often disabled by default
- Example: a previous employer

WEP key size and IV size

- Original sizes: 40-bit shared key (export restrictions) plus 24-bit IV = 64-bit RC4 key
  - Both too small
- 128-bit upgrade kept 24-bit IV
  - Vague about how to choose IVs
  - Least bad: sequential, collision takes hours
  - Worse: random or everyone starts at zero

WEP RC4 related key attacks

- Only true crypto weakness
- RC4 "key schedule" vulnerable when:
  - RC4 keys very similar (e.g., same key, similar IV)
  - First stream bytes used
- Not a practical problem for other RC4 users like SSL
  - Key from a hash, skip first output bytes

More recent problem with WPA (CCS'17)

- Session key set up in a 4-message handshake
- Key reinstallation attack: replay #3
  - Causes most implementations to reset nonce and replay counter
  - In turn allowing many other attacks
  - One especially bad case: reset key to 0
- Protocol state machine behavior poorly described in spec
  - Outside the scope of previous security proofs

Trustworthiness of primitives

- Classic worry: DES S-boxes
- Obviously in trouble if cipher chosen by your adversary
- In a public spec, most worrying are unexplained elements
- Best practice: choose constants from well-known math, like digits of π

Dual_EC_DRBG (1)

- Pseudorandom generator in NIST standard, based on elliptic curve
- Looks like provable (slow enough!) but strangely no proof
- Specification includes long unexplained constants
- Academic researchers find:
  - Some EC parts look good
  - But outputs are statistically distinguishable

Dual_EC_DRBG (2)

- Found 2007: special choice of constants allows prediction attacks
  - Big red flag for paranoid academics
- Significant adoption in products sold to US govt. FIPS-140 standards
  - Semi-plausible rationale from RSA (EMC)
- NSA scenario basically confirmed by Snowden leaks
  - NIST and RSA immediately recommend withdrawal

Outline

DNS: trusted but vulnerable

- Almost every higher-level service interacts with DNS
- UDP protocol with no authentication or crypto
  - Lots of attacks possible
- Problems known for a long time, but challenge to fix compatibly

DNSSEC goals and non-goals

+ Authenticity of positive replies
+ Authenticity of negative replies
+ Integrity
− Confidentiality
− Availability

First cut: signatures and certificates

- Each resource record gets an `RRSIG` signature
  - E.g., `A` record for one name→address mapping
  - Observe: signature often larger than data
- Signature validation keys in `DNSKEY` RRs
- Recursive chain up to the root (or other "anchor")

Add more indirection

- DNS needs to scale to very large flat domains like `.com`
- Facilitated by having single `DS` RR in parent indicating delegation
- Chain to root now includes `DS`es as well

Negative answers

- Also don't want attackers to spoof non-existence
  - Gratuitous denial of service, force fallback, etc.
- But don't want to sign "x does not exist" for all x
- Solution 1, `NSEC`: "there is no name between `acacia` and `baobab`"

Preventing zone enumeration

- Many domains would not like people enumerating all their entries
- DNS is public, but "not that public"
- Unfortunately `NSEC` makes this trivial
- Compromise: `NSEC3` uses password-like salt and repeated hash, allows opt-out

DANE: linking TLS to DNSSEC

- "DNS-based Authentication of Named Entities"
- DNS contains hash of TLS cert, don't need CAs
- How is DNSSEC's tree of certs better than TLS's?

Signing the root

- Political problem: many already distrust US-centered nature of DNS infrastructure
- Practical problem: must be very secure with no single point of failure
- Finally accomplished in 2010
  - Solution involves 'key ceremonies', international committees, smart cards, safe deposit boxes, etc.

Deployment

- Standard deployment problem: all cost and no benefit to being first mover
- Servers working on it, mostly top-down
- Clients: still less than 20%
- Will probably be common for a while: insecure connection to secure resolver

What about privacy?

- Users increasingly want privacy for their DNS queries as well
- Older DNSCurve and DNSCrypt protocols were not standardized
- More recent "DNS over TLS" and "DNS over HTTPS" are RFCs
- DNS over HTTPS in major browsers might have serious centralization effects