CSci 4271W
Development of Secure Software Systems
Day 19: Web part 3 and cryptography part 1

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

Third party content / web bugs

- Much tracking involves sites other than the one in the URL bar
  - For fun, check where your cookies are coming from
- Various levels of cooperation
- *Web bugs* are typically 1x1 images used only for tracking

Cookies arms race

- Privacy-sensitive users like to block and/or delete cookies
- Sites have various reasons to retain identification
- Various workarounds:
  - Similar features in Flash and HTML5
  - Various channels related to the cache
  - *Evercookie*: store in n places, regenerate if subset are deleted

Browser fingerprinting

- Combine various server or JS-visible attributes passively
  - User agent string (10 bits)
  - Window/screen size (4.83 bits)
  - Available fonts (13.9 bits)
  - Plugin verions (15.4 bits)

(Data from `panopticlick.eff.org`, far from exhaustive)

History stealing

- History of what sites you've visited is not supposed to be JS-visible
- But, many side-channel attacks have been possible
  - Query link color
  - CSS style with external image for visited links
  - Slow-rendering timing channel
  - Harvesting bitmaps
  - User perception (e.g. fake CAPTCHA)

Browser and extension choices

- More aggressive privacy behavior lives in extensions
  - Disabling most JavaScript (NoScript)
  - HTTPS Everywhere (centralized list)
  - Tor Browser Bundle
- Default behavior is much more controversial
  - Concern not to kill advertising support as an economic model

Outline

Misconfiguration problems

- Default accounts
- Unneeded features
- Framework behaviors
  - Don't automatically create variables from query fields

Openness tradeoffs

- Error reporting
  - Few benign users want to see a stack backtrace
- Directory listings
  - Hallmark of the old days
- Readable source code of scripts
  - Doesn't have your DB password in it, does it?

Using vulnerable components

- Large web apps can use a lot of third-party code
- Convenient for attackers too
  - OWASP: two popular vulnerable components downloaded 22m times
- Hiding doesn't work if it's popular
- Stay up to date on security announcements

Clickjacking

- Fool users about what they're clicking on
  - Circumvent security confirmations
  - Fabricate ad interest
- Example techniques:
  - Frame embedding
  - Transparency
  - Spoof cursor
  - Temporal "bait and switch"

Crawling and scraping

- A lot of web content is free-of-charge, but proprietary
  - Yours in a certain context, if you view ads, etc.
- Sites don't want it downloaded automatically (*web crawling*)
- Or parsed and user for another purpose (*screen scraping*)
- High-rate or honest access detectable

Outline

Confidentiality and privacy, cont'd

Even more web risks

Announcements intermission

Crypto basics

Stream ciphers

Course reminders

- The OWASP Top Ten reading quiz is due tonight
- Project 1 submission 1's regular deadline is Friday night
  - Please bring more questions to office hours and Piazza

Non-course reminders

- Today is Election Day; in Minneapolis, it is the city council election
- Polls are open until 8pm tonight

Outline

- Confidentiality and privacy, cont'd
- Even more web risks
- Announcements intermission
- Crypto basics
- Stream ciphers

-ography, -ology, -analysis

- Cryptography (narrow sense): designing encryption
- Cryptanalysis: breaking encryption
- Cryptology: both of the above
- Code (narrow sense): word-for-concept substitution
- Cipher: the "codes" we actually care about

Caesar cipher

- Advance three letters in alphabet: $A \to D, B \to E, \ldots$
- Decrypt by going back three letters
- Internet-era variant: rot-13
- Easy to break if you know the principle

Keys and Kerckhoffs's principle

- The only secret part of the cipher is a *key*
- Security does not depend on anything else being secret
- Modern (esp. civilian, academic) crypto embraces openness quite strongly

Symmetric vs. public key

- Symmetric key (today's lecture): one key used by all participants
- Public key: one key kept secret, another published
  - Techniques invented in 1970s
  - Makes key distribution easier
  - Depends on fancier math

Goal: secure channel

- Leaks no content information
  - Not protected: size, timing
- Messages delivered intact and in order
  - Or not at all
- Even if an adversary can read, insert, and delete traffic

One-time pad

- Secret key is truly random data as long as message
- Encrypt by XOR (more generally addition mod alphabet size)
- Provides perfect, "information-theoretic" secrecy
- No way to get around key size requirement

Computational security

- More realistic: assume adversary has a limit on computing power
- Secure if breaking encryption is computationally infeasible
  - E.g., exponential-time brute-force search
- Ties cryptography to complexity theory

Key sizes and security levels

- Difficulty measured in powers of two, ignore small constant factors
- Power of attack measured by number of steps, aim for better than brute force
- 2^{32} definitely too easy, probably 2^{64} too
- Modern symmetric key size: at least 2^{128}

Crypto primitives

- Base complicated systems on a minimal number of simple operations
- Designed to be fast, secure in wide variety of uses
- Study those primitives very intensely

Attacks on encryption

- Known ciphertext
  - Weakest attack
- Known plaintext (and corresponding ciphertext)
- Chosen plaintext
- Chosen ciphertext (and plaintext)
  - Strongest version: adaptive

Certificational attacks

- Good primitive claims no attack more effective than brute force
- Any break is news, even if it's not yet practical
  - Canary in the coal mine
- E.g., $2^{126.1}$ attack against AES-128
- Also watched: attacks against simplified variants

Fundamental ignorance

- We don't really know that any computational cryptosystem is secure
- Security proof would be tantamount to proving $\mathrm{P} \neq \mathrm{NP}$
- Crypto is fundamentally more uncertain than other parts of security

Relative proofs

- Prove security under an unproved assumption
- In symmetric crypto, prove a construction is secure if the primitive is
  - Often the proof looks like: if the construction is insecure, so is the primitive
- Can also prove immunity against a particular kind of attack

Random oracle paradigm

- Assume ideal model of primitives: functions selected uniformly from a large space
  - Anderson: elves in boxes
- Not theoretically sound; assumption cannot be satisfied
- But seems to be safe in practice

Pseudorandomness and distinguishers

- Claim: primitive cannot be distinguished from a truly random counterpart
  - In polynomial time with non-negligible probability
- We can build a distinguisher algorithm to exploit any weakness
- Slightly too strong for most practical primitives, but a good goal

Open standards

- How can we get good primitives?
- Open-world best practice: run competition, invite experts to propose then attack
- Run by neutral experts, e.g. US NIST
- Recent good examples: AES, SHA-3

A certain three-letter agency

- National Security Agency (NSA): has primary responsibility for "signals intelligence"
- Dual-mission tension:
  - Break the encryption of everyone in the world
  - Help US encryption not be broken by foreign powers

Outline

Confidentiality and privacy, cont'd

Even more web risks

Announcements intermission

Crypto basics

Stream ciphers

Stream ciphers

- Closest computational version of one-time pad
- Key (or seed) used to generate a long pseudorandom bitstream
- Closely related: cryptographic RNG

Shift register stream ciphers

- Linear-feedback shift register (LFSR): easy way to generate long pseudorandom sequence
  - But linearity allows for attack
- Several ways to add non-linearity
- Common in constrained hardware, poor security record

RC4

- Fast, simple, widely used software stream cipher
  - Previously a trade secret, also "ARCFOUR"
- Many attacks, none yet fatal to careful users (e.g. TLS)
  - Famous non-careful user: WEP
- Now deprecated, not recommended for new uses

Encryption $\neq$ integrity

- Encryption protects secrecy, not message integrity
- For constant-size encryption, changing the ciphertext just creates a different plaintext
- How will your system handle that?
- Always need to take care of integrity separately

Stream cipher mutability

- Strong example of encryption vs. integrity
- In stream cipher, flipping a ciphertext bit flips the corresponding plaintext bit, only
- Very convenient for targeted changes

Salsa and ChaCha

- Published by Daniel Bernstein 2007-2008
- Stream cipher with random access to stream
  - Related to counter mode discussed later
- Fast on general-purpose CPUs without specialized hardware
- Adopted as option for TLS and SSH
  - Prominent early adopter: Chrome on Android

Stream cipher assessment

- Currently less fashionable as a primitive in software
- Not inherently insecure
  - Other common pitfall: must not reuse key(stream)