### CSci 4271W Development of Secure Software Systems Day 15: Web Application Security, part 3

Stephen McCamant University of Minnesota, Computer Science & Engineering

### Outline

### SQL injection

Injection attack demo

- Confidentiality and privacy
- Even more web risks
- Crypto basics

### Relational model and SQL

- Relational databases have tables with rows and single-typed columns
- Used in web sites (and elsewhere) to provide scalable persistent storage
- Allow complex *queries* in a declarative language SQL

### **Example SQL queries**

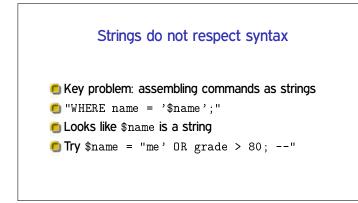
- SELECT name, grade FROM Students WHERE grade < 60 ORDER BY name;</p>
- UPDATE Votes SET count = count + 1 WHERE candidate = 'John';

### Template: injection attacks

 Your program interacts with an interpreted language
 Untrusted data can be passed to the interpreter
 Attack data can break parsing assumptions and execute arbitrary commands

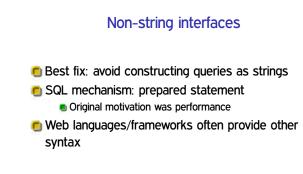
### SQL + injection

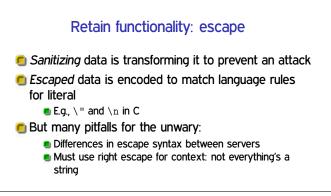
- Why is this named most critical web app. risk?
- Easy mistake to make systematically
- Can be easy to exploit
- Database often has high-impact contents E.g., logins or credit cards on commerce site



### Using tautologies

- 🖲 Tautology: formula that's always true
- Often convenient for attacker to see a whole table
- Classic: OR 1=1





### Lazy sanitization: allow-listing

- Allow only things you know to be safe/intended
- 🖲 Error or delete anything else
- Short allow-list is easy and relatively easy to secure
- E.g., digits only for non-negative integer
- But, tends to break benign functionality

### Poor idea: deny-listing

- Space of possible attacks is endless, don't try to think of them all
- Want to guess how many more comment formats SQL has?
- Particularly silly: deny 1=1

### Attacking without the program

### Often web attacks don't get to see the program Not even binary, it's on the server

- Surmountable obstacle:
  - Guess natural names for columns
  - Harvest information from error messages

### Blind SQL injection

- Attacking with almost no feedback
- Common: only "error" or "no error"
- One bit channel you can make yourself: if (x) delay 10 seconds
- Trick to remember: go one character at a time

### Injection beyond SQL

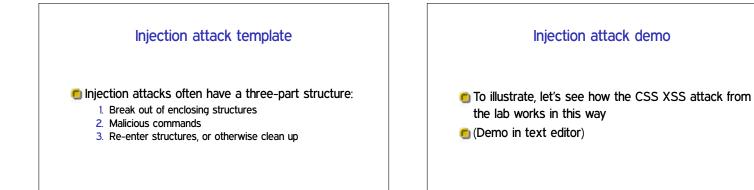
Shell commands, format strings, XSS
 XPath/XQuery: queries on XML data
 LDAP: queries used for authentication

### Outline

### SQL injection

Injection attack demo

- Confidentiality and privacy
- Even more web risks
- **Crypto basics**



Outline

SQL injection

**Crypto basics** 

Injection attack demo

Even more web risks

Confidentiality and privacy

Consider disabling autocomplete

Consider disabling caching

Performance tradeoff

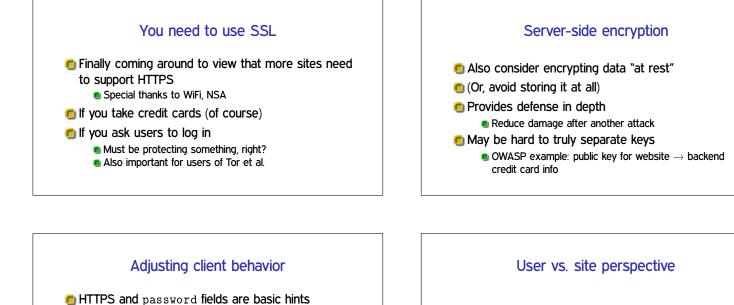
Or proxy? You need SSL

Finally standardized in HTML5

Better not to have this on user's disk

Usability tradeoff, save users from themselves

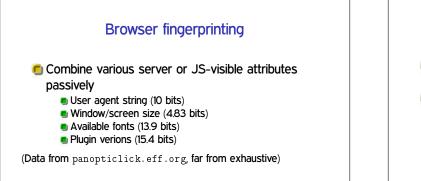




- User privacy goals can be opposed to site goals
- Such as in tracking for advertisements
- Browser makers can find themselves in the middle Of course, differ in institutional pressures







### 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

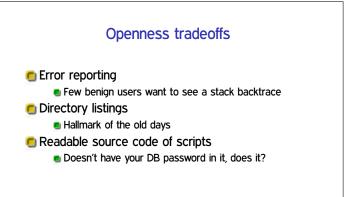
- Disabling most JavaScript (NoScript)
- HTTPS Everywhere (centralized list)
- Tor Browser Bundle
- Default behavior is much more controversial
  - Concern not to kill advertising support as an economic model

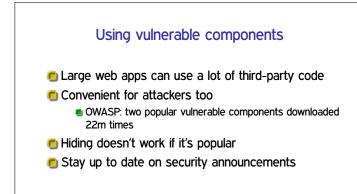
### Outline

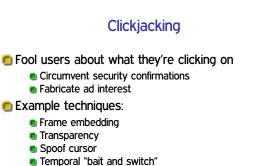
### SQL injection

- Injection attack demo
- Confidentiality and privacy
- Even more web risks
- **Crypto basics**

## Misconfiguration problems Default accounts Unneeded features Framework behaviors Don't automatically create variables from query fields







## 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



SQL injection

Injection attack demo

- Confidentiality and privacy
- Even more web risks

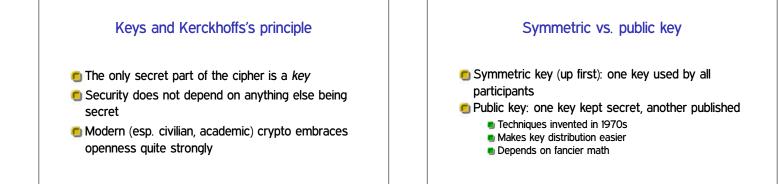
Crypto basics

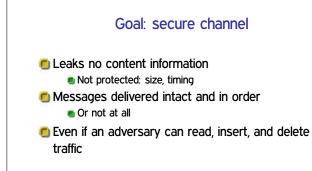
### -ography, -ology, -analysis

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





### 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



- 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
- $\bigcirc 2^{32}$  definitely too easy, probably  $2^{64}$  too
- Modern symmetric key size: at least 2<sup>128</sup>

### 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

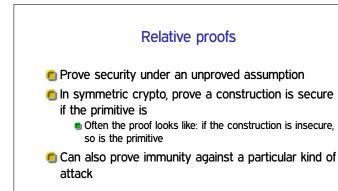
### Attacks on encryption

- 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<sup>126.1</sup> 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  $P \neq NP$
- Crypto is fundamentally more uncertain than other parts of security



### 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:

- Dual-MISSION LENSION.
  - Break the encryption of everyone in the world
  - Help US encryption not be broken by foreign powers