CONSTRUCTOR UNIVERSITY

Security and Authorization

Ramakrishnan & Gehrke, Chapter 21


NAUTOMAT


YOU REMEMBER MY PIN?

Overview

- Introduction
- Internet security
- Database access control
- How to hack a database

Intro to DB & Web Service Security

- Secrecy:
  Users should not be able to see things they are not supposed to
  - Ex: student can't see other students' grades
- Ex: *TJX*. owns many dept stores in US
  - Attacks exploited WEP used at branches
  - Over 47 million CC #s stolen dating back to 2002
  - *…sue filed by consortium of 300 banks*
- Ex: *CardSystems, Inc:* US credit card payment processing company
  - 263,000 CC #s stolen from database via SQL injection (June 2005)
  - 43 million CC #s stored unencrypted, compromised
  - *…out of business*

Intro to DB & Web Service Security

- Secrecy:
  Users should not be able to see things they are not supposed to
  - Ex: student can't see other students' grades
- Ex: *Equifax 2017* [Siliconbeat]
  - Collecting most sensitive citizen data for credit assessment
    - ssn, name, address, birth dates, credit cards, driver's license, history, …
    - 143m customers affected
  - "maybe dozens" of breaches, fix only 6 months after warning
  - hacked due to insufficient internal security; patch not installed, but got known
  - *BTW, senior execs sold 1.8m in stock*

It would be nice to think that perhaps the company was a victim […] of clever hackers using social engineering […], but it appears […] that there is gross incompetence involved.

Intro to DB & Web Service Security

- Secrecy:
  Users should not be able to see things they are not supposed to
  - Ex: student can't see other students' grades
- Integrity:
  Users should not be able to modify things they are not supposed to
  - Ex: Only instructors can assign grades
- Availability:
  Users should be able to see and modify things they are allowed to
  - Ex: professor can see and set students' grades(but possibly not modify after release)

UK GCHQ Manipulating Internet [src]

- “Change outcome of online polls” (UNDERPASS)
- “Disruption of video-based websites hosting extremist content through concerted target discovery and content removal.” (SILVERLORD)
- “Active skype capability. Provision of real time call records (SkypeOut and SkypetoSkype) and bidirectional instant messaging. Also contact lists.” (MINIATURE HERO)
- “Find private photographs of targets on Facebook” (SPRING BISHOP)
- “Permanently disable a target’s account on their computer” (ANGRY PIRATE)
- “Targeted Denial Of Service against Web Servers” (PREDATORS FACE)
- “Monitoring target use of the UK eBay” (ELATE)
- “Spoof any email address and send email under that identity” (CHANGELING)
- ...

“If you don’t see it here, it doesn’t mean we can’t build it.”

Overview

- Introduction
- Internet security
- Database access control
- How to hack a database

Internet-Oriented Security

- Key Issues: User authentication and trust
  - For DB access from secure location, password-based schemes usually adequate
- For access over an external network, trust is hard to achieve
  - If someone with Sam's credit card wants to buy from you, how can you be sure it is not someone who stole his card?
  - How can Sam be sure that the screen for entering his credit card information is indeed yours, and not some rogue site spoofing you (to steal such information)?
  - How can he be sure that sensitive information is not "sniffed" while it is being sent over the network to you?
- Encryption is a technique used to address these issues

Encryption

- Idea: “Mask” data for secure transmission or storage
  - Encrypt(data, encryption key) = encrypted data
  - Decrypt(encrypted data, decryption key) = original data
- Symmetric Encryption: DES (Data Encryption Standard)
  - Encryption key = decryption key → all authorized users know decryption key
  - DES (since 1977) 56-bit key; AES 128-bit (or 192-bit or 256-bit) key
  - 1024-bit key considered relatively safe, 2048 preferred
- Public-Key Encryption: Each user has two keys (RSA, Turing Award)
  - User’s encryption key: public
  - User’s decryption key: secret

Certifying Servers

- Amazon distributes their public key, Sam's browser encrypts order using it
  - So, only Amazon can decipher the order, since no one else has Amazon's private key
- SSL protocol to know that public key for Amazon is genuine
  - Amazon contracts with Verisign → certificate <Verisign,Amazon,amazon.com,*public-key*>
  - stored encrypted with Verisign's private key, known only to Verisign
  - Verisign's public key known to all browsers, can therefore decrypt certificate and obtain Amazon's public key, and be confident that it is genuine
  - browser generates temp session key, encodes it using Amazon's public key, sends to Amazon
  - All subsequent messages between the browser and Amazon are encoded using symmetric encryption (e.g., DES), which is more efficient than public-key encryption
- What if Sam doesn't trust Amazon with his credit card information?
  - Secure Electronic Transaction (SET) protocol:
    3-way communication between Amazon, Sam, and trusted server, e.g., Visa

Authenticating Users With SSL

- Amazon can simply use password authentication
  - Sam logs into Amazon account; establishes session key via SSL → pw transmission secure (?)
  - Amazon still at risk if Sam's card stolen + password hacked. Business risk …
- Digital Signatures:
  - Sam encrypts order using his private key, then encrypts result using Amazon's public key
  - Amazon decrypts msg with their private key, decrypts result using Sam's public key, yields original order!
  - Exploits interchangeability of public/private keys for encryption/decryption
  - Now, no one can forge Sam's order, and Sam cannot claim that someone else forged the order

Setting Up Private & Public Keys

- https = http over secure socket layer (SSL), using port 443
  - https://www.example.com/…
  - OpenSSL: open-source SSL library
- Create a private key
  - openssl genrsa -out {rsa.private} 1024
- Create self-signed certificate
  - openssl rsa -in {rsa.private} -out {rsa.public} -pubout -outform PEM
  - browser will pop up a warning, since it cannot identify server, but communication still encrypted
    - To stop this, buy an official SSL certificate
      → Verisign, Thawte,… ($200/year upwards) or letsencrypt (free)

Configure Server for https

- Add to /etc/httpd/httpd.conf:

```
<http port="443">
   <openssl>
      <certificate-file>keys/my.crt</certificate-file>
      <certificate-key-file>keys/my.key</certificate-key-
 file>
      <password>my-password</password>
   </openssl>
</http>
```

- ready to go (RTFM for your system)

1. Email Security

- Classic way to achieve security: email disclaimers
  - Standard legalese: "*This message is confidential. It may also be privileged or otherwise protected by work product immunity or other legal rules. If you have received it by mistake, please let us know by e-mail reply and delete it from your system; you may not copy this message or disclose its contents to anyone. Please send us by fax any message containing deadlines as incoming e-mails are not screened for response deadlines. The integrity and security of this message cannot be guaranteed on the Internet.*"
  - BTW, oldest found (AD 1083): "*Si forte in alienas manus oberraverit hec peregrina epistola incertis ventis dimissa, sed Deo commendata, precamur ut ei reddatur cui soli destinata, nec preripiat quisquam non sibi parata*."
- Compare to a paper letter..
- PS: I like this one: http://www.goldmark.org/jeff/stupid-disclaimers

1. Email Security / contd.

- "…mostly, legally speaking, pointless. Lawyers and experts on internet policy say no court case has ever turned on the presence or absence of such an automatic e-mail footer in America, the most litigious of rich countries."
  - But, comment:
    „They are prevalent in the U.S. exactly BECAUSE there is no court case that has turned on the appearance or lack of a disclaimer or end of email boiler plate. Until a court affirmatively denies their power, they will remain […]."
- "Many disclaimers are, in effect, seeking to impose a contractual obligation unilaterally, and thus are probably unenforceable. This is clear in Europe."
- [lifehacker.com]

Disclaimer: this is not a legal advice, I'm not a lawyer. No responsibility whatsoever taken

1. Email Security / contd.

[George Merticariu]

- Risks to user
  - Disclosure of Information by plain text transmission
    - *Traffic analysis: in some countries emails monitored by agencies*
  - Modification: "man-in-the-middle attack"
  - Masquerade: send in the name of others
  - Denial of Service: overloading servers; blocking users by repeatedly wrong password
- Safe email = encryption + signature
  - PGP (Pretty Good Privacy), SecureGmail, …
  - Ex: `gpg --list-secret-keys` `gpg --sign {myfile}`
  - mailers support encryption today – find out how to enable!

2. Web Applications Security

- Unauthorized: No way!
- User / password transmission: risky
- API Security without credentials: application signature + token
  - initial call: get verification token (and its lifetime)
  - Next calls: send token + signature for validation,
    new token generated,
    old token expires
  - application signature generated from “application secret” + timestamp + other data
    →need to know algorithm + the secret

Overview

- Introduction
- Internet security
- Database access control
- How to hack a database

Database Access Control

- A security policy specifies who is authorized to do what
  - Authentication: “let me in”
  - Authorization: “let me do this”
- A security mechanism allows us to enforce a chosen security policy
  - Implemented in DBMS
- Two main mechanisms at DBMS level:
  - Discretionary access control (=security at users’ discretion)
  - Mandatory access control (=security enforced)

Discretionary Access Control

- concept of access rights or privileges for objects (tables and views), and mechanisms for giving users privileges (and revoking privileges)
- Creator of a table or a view automatically gets all privileges on it
  - DMBS keeps track of who subsequently gains & loses privileges
  - Allows only requests from users with necessary privileges (at request time)

GRANT Command

GRANT privileges ON object TO users [WITH GRANT OPTION]

- The following privileges can be specified:
  - SELECT: Can read all columns (incl those added later via ALTER TABLE command)
  - INSERT(col-name): Can insert tuples with non-null or non-default values
    - *INSERT means same right with respect to all columns*
  - DELETE: Can delete tuples
  - REFERENCES (col-name): Can define foreign keys (other tables) to this column
- If a user has a privilege with the GRANT OPTION, can pass privilege on to other users (with or without passing on the GRANT OPTION)
- Only owner can execute CREATE, ALTER, and DROP

GRANT and REVOKE of Privileges

- GRANT INSERT, SELECT ON Sailors TO Horatio
  - Horatio can query Sailors or insert tuples into it
- GRANT DELETE ON Sailors TO Yuppy WITH GRANT OPTION
  - Yuppy can delete tuples, and also authorize others to do so
- GRANT UPDATE (rating) ON Sailors TO Dustin
  - Dustin can update (only) the rating field of Sailors tuples
- GRANT SELECT ON ActiveSailors TO Guppy, Yuppy
  - This does NOT allow the 'uppies to query Sailors directly!
- REVOKE cascades: When a privilege is revoked from X, it is also revoked from all users who got it solely from X

Views and Security

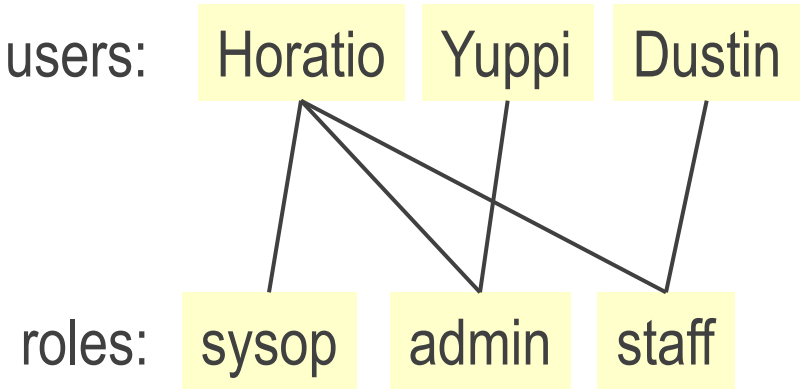
- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s)
  - Given ActiveSailors, but not Sailors or Reserves, we can find sailors who have a reservation, but not the bid's of boats that have been reserved
- Creator of view has a privilege on the view if (s)he has the privilege on all underlying tables
- Together with GRANT/REVOKE commands, views are a very powerful access control tool

Security to the Level of a Field!

- Can create a view that only returns one field of one tuple (How?)
- Then grant access to that view accordingly
- Allows for arbitrary granularity of control, but:
  - Clumsy to specify, though this can be hidden under a good UI
  - Performance is unacceptable if we need to define field-granularity access frequently
    - *Too many view creations and look-ups*

Role-Based Authorization

- In SQL-92, privileges were actually assigned to authorization ids, which can denote a single user or a group of users
- In SQL:1999 (and in many current systems), privileges are assigned to roles
  - Roles can then be granted to users and to other roles
  - Reflects how real organizations work
  - Illustrates how standards often catch up with “de facto” standards embodied in popular systems



Mandatory Access Control

- Based on system-wide policies that cannot be changed by individual users
  - Each DB object is assigned a security class
  - Each subject (user or user program) is assigned a clearance for a security class
  - Rules based on security classes and clearances govern who can read/write which objects
- Most commercial systems do not support mandatory access control
- Versions of some DBMSs do support it; used for specialized (e.g., defence) applications

Why Mandatory Control?

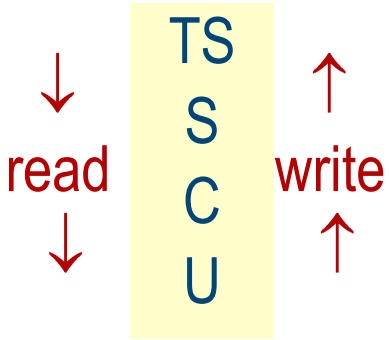
- Discretionary control has some flaws, e.g., the Trojan horse problem:
  - Dick creates table Horsie and gives INSERT privileges to Justin (who doesn't know about this)
  - Dick modifies the code of an application program used by Justin to additionally write some secret data to table Horsie
  - Now, Dick can see the secret data
- modification of code is beyond DBMS's control,
  but can prevent use of the database as channel for secret information

Bell-LaPadula Model

- Objects
  - tables, views, tuples, …
- Subjects
  - users, user programs, …
- Security classes: TS > S> C > U
  - Top secret (TS), secret (S), confidential (C), unclassified (U)
- Each object and subject is assigned a class
  - Simple Security Property:
    Subject S can read object O only if class(S) >= class(O))
  - *-Property:
    Subject S can write object O only if class(S) <= class(O)

Bell-LaPadula Model: Intuition

- Idea: ensure that
  - information can only be read
    from higher to lower security levels
  - Information can only be written
    from lower to higher security level

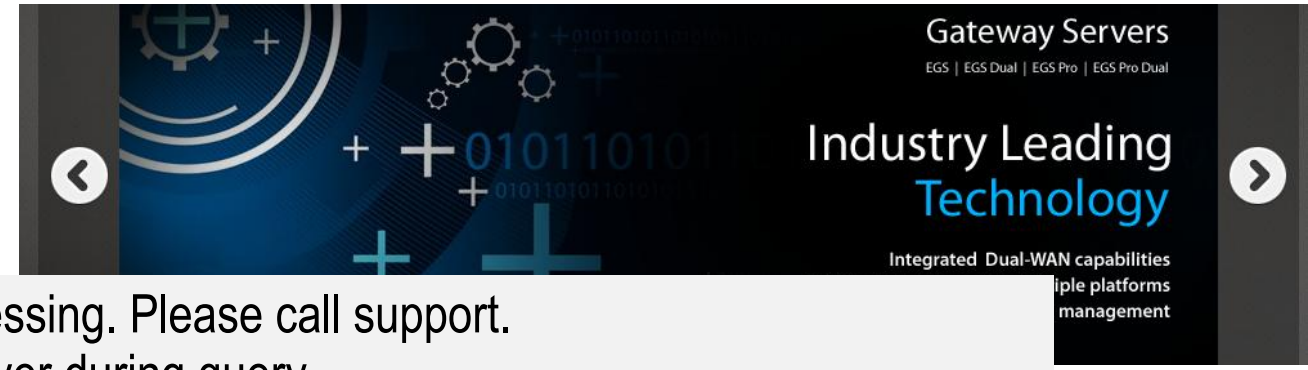


- Ex: Dick has security class C,
  Justin has class S, secret table has class S
  - Dick's table, Horsie, has Dick's clearance, C
  - Justin's application has his clearance, S
  - So, Justin's program (modified by Dick) cannot write into table Horsie
- Mandatory access control rules are applied in addition to any discretionary controls that are in effect

Overview

- Introduction
- Internet security
- Database access control
- How to hack a database

How to Expose Yourself

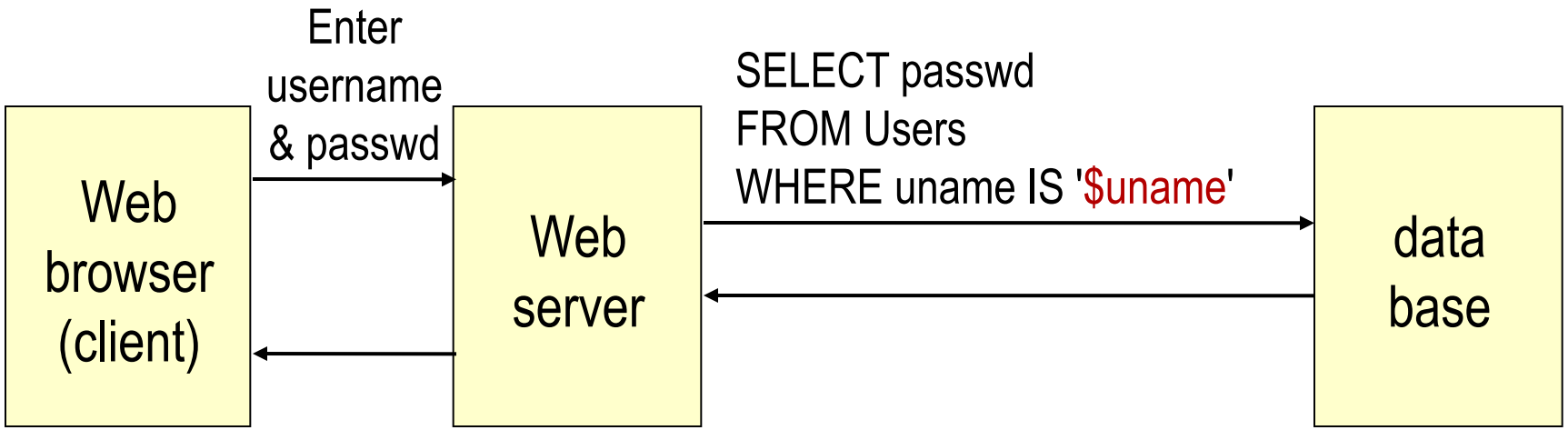


An error occured durring processing. Please call support.
Lost connection to MySQL server during query
SQL: select count(*) from LoginsActive where MacAddress=\'00:21:70:6E:04:AE\' and MacAddress!=\'\' and Iface=\'br0\' and PropertyID=\'51225\'
IP:sql.ethostream.com
DBU:remote
DB:

OK, that was in 2011.

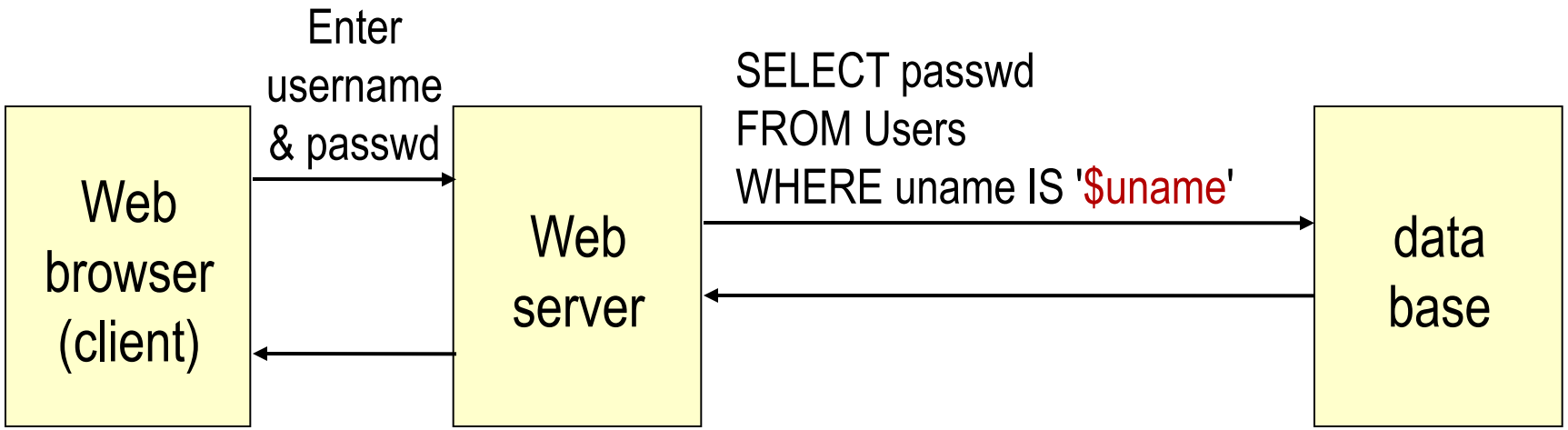
How To Hack a Database

- Most common: SQL injection
  - Compromise database query



How To Hack a Database (contd.)

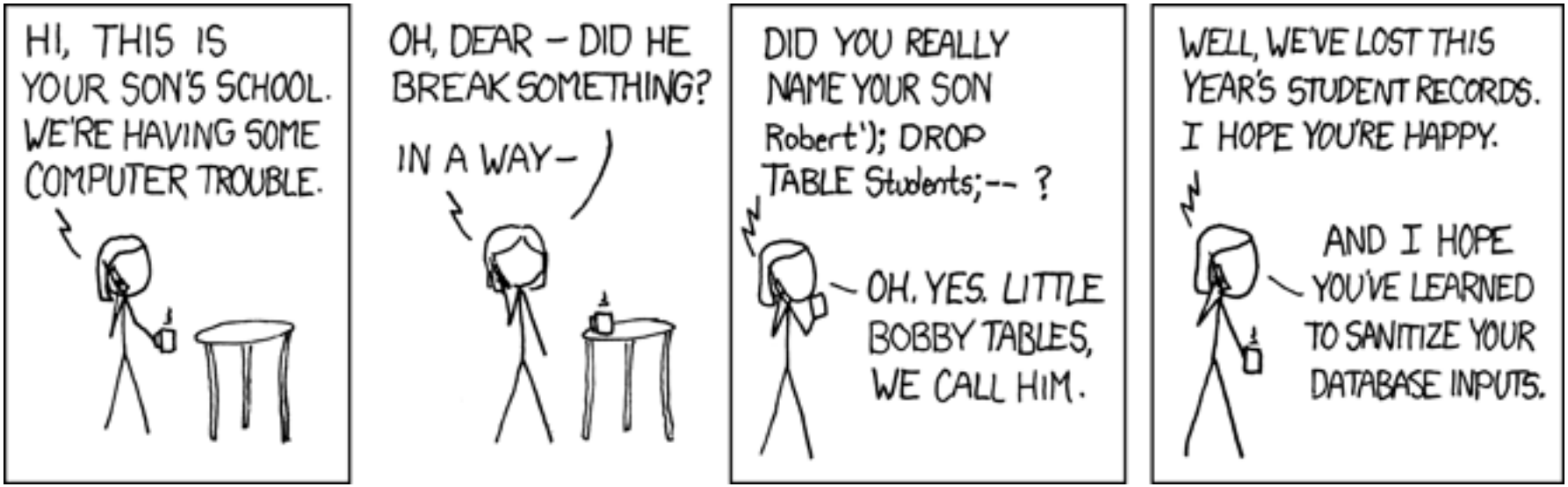
- Most common: SQL injection
  - Compromise database query



- What will happen at input of '; DROP TABLE Users; -- ? (keyword: DoS)
- *Name 2 independent techniques to prevent!*

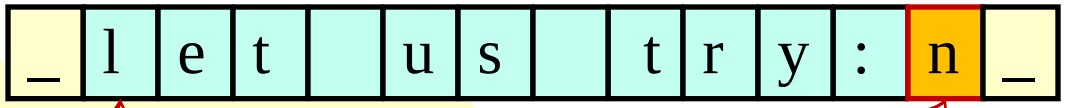
Mom 's a Hacker

[found by: Prashant Vaibhav]



Hacking, Generalized

- SQL injection generalizes to: Command injection
  - ...usually by abusing data paths as command paths
- Ex: buffer overflow attack

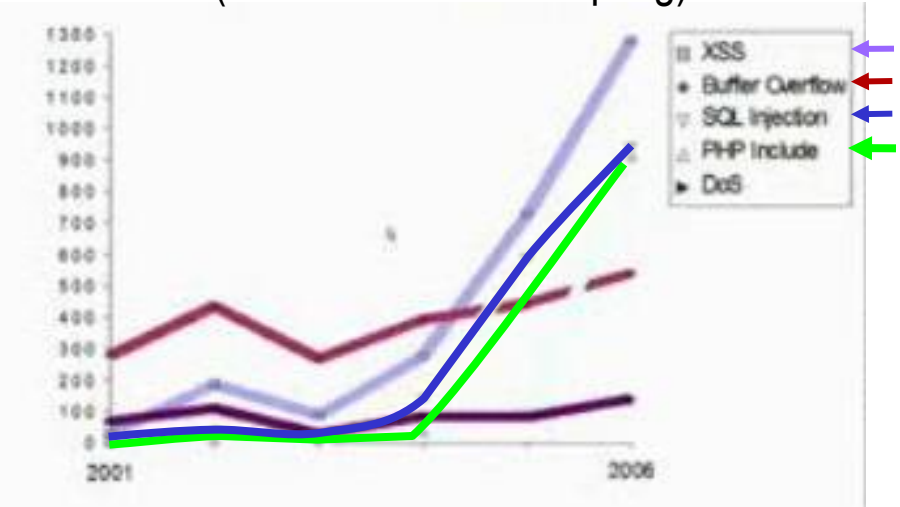



```
{    char inputData[11];
     char command;
     switch (command)
     {    case `s`: executeSelect( inputData ); break;
          case `u`: executeUpdate( inputData ); break;
          case `i`: executeInsert( inputData ); break;
          case `d`: executeDelete( inputData ); break;
          case `n`: detonateNuke(); break;
     }
}
```

SW Reasons for Service Attacks

- Missing input validation
- Design errors
- Boundary conditions
- Exception handling
- Access validation

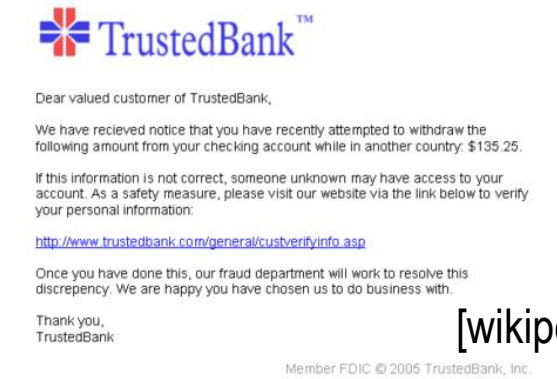
Vulnerability trends [Mitre]
(XSS = cross-site scripting)



- *Red = targets with increasing stats*
  - *See also: OWASP Top 10*

Common Internet Attacks

- spear-phishing
  - = acquire information (usernames, passwords, CC details, …) by masquerading as a trustworthy entity
- man in the middle (→ eavesdropping)
  - = attacker makes independent connections with victims, relays messages between them → victims believe they talk directly to each other
  - attacker intercepts all messages + injects new ones
- watering-hole
  - = attack group:
    Guess / observe sites which group often uses;
    infect these; eventually, some will get infected.


TrustedBank™

Dear valued customer of TrustedBank,

We have recieved notice that you have recently attempted to withdraw the following amount from your checking account while in another country: $135.25.

If this information is not correct, someone unknown may have access to your account. As a safety measure, please visit our website via the link below to verify your personal information:

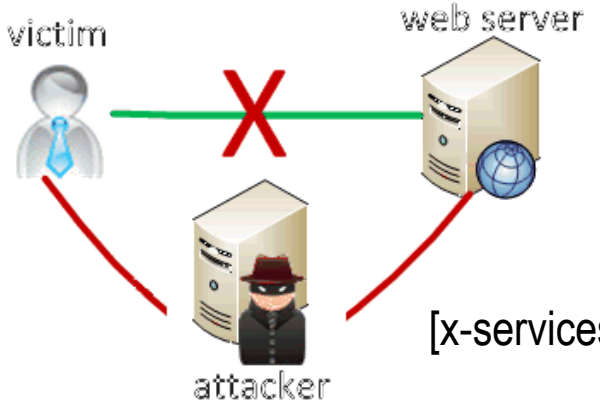
http://www.trustedbank.com/general/custverifyinfo.asp

Once you have done this, our fraud department will work to resolve this discrepency. We are happy you have chosen us to do business with.

Thank you,
TrustedBank

Member FDIC © 2005 TrustedBank, Inc.

[wikipedia]



[x-services.nl]

Biggest Identity Leak to Date

- Discovered by Hold Security,
  reported in the New York times (Aug 5, 2014)
- 420,000 websites compromised,
  1.2 billion user password data, 500 million e-mail addresses
- presumably bots carrying out automated SQL injection attacks

- PS: https://sec.hpi.uni-potsdam.de/leak-checker/

Case Study: Common Security Neglicences

CONSTRUCTOR UNIVERSITY

- In 2014, Sony Pictures suffered major break-in
  - possibly by North Korea, in relation to movie *The Interview*
  - "mostly facilitated by unprecedented negligence"
- Problems included:
  - unencrypted storage of sensitive information
  - password stored in plain text files, sometimes even called "passwords" or placed in same directory as encrypted files
  - easily guessable passwords
  - large number of unmonitored devices
  - lack of accountability and responsibility for security, ignorance towards recommendations and audits
  - lack of systematic lesson-learning from previous failures (which included 2011 hacks of Sony PlayStation Network and Sony Pictures that stole account information including unsalted or plain text passwords)
  - weak IT and information security teams

„salted" ?

- Stolen data included employee data (including financial data), internal emails, and movies

Afterthoughts: Security and Software Engineering

- Additional security related engineering principles, such as: [Neil Daswani]
  - least privilege
    - *No more rights for any app than absolutely necessary*
  - fail-safe stance
    - *Always return to safe, stable state, after any kind of deviation*
  - protecting against weakest link
    - *Rank vulnerability of components, pay particular attention to "champions"*

mobile apps?

- 3 P security management:
  Process,
  People,
  Probing your defences