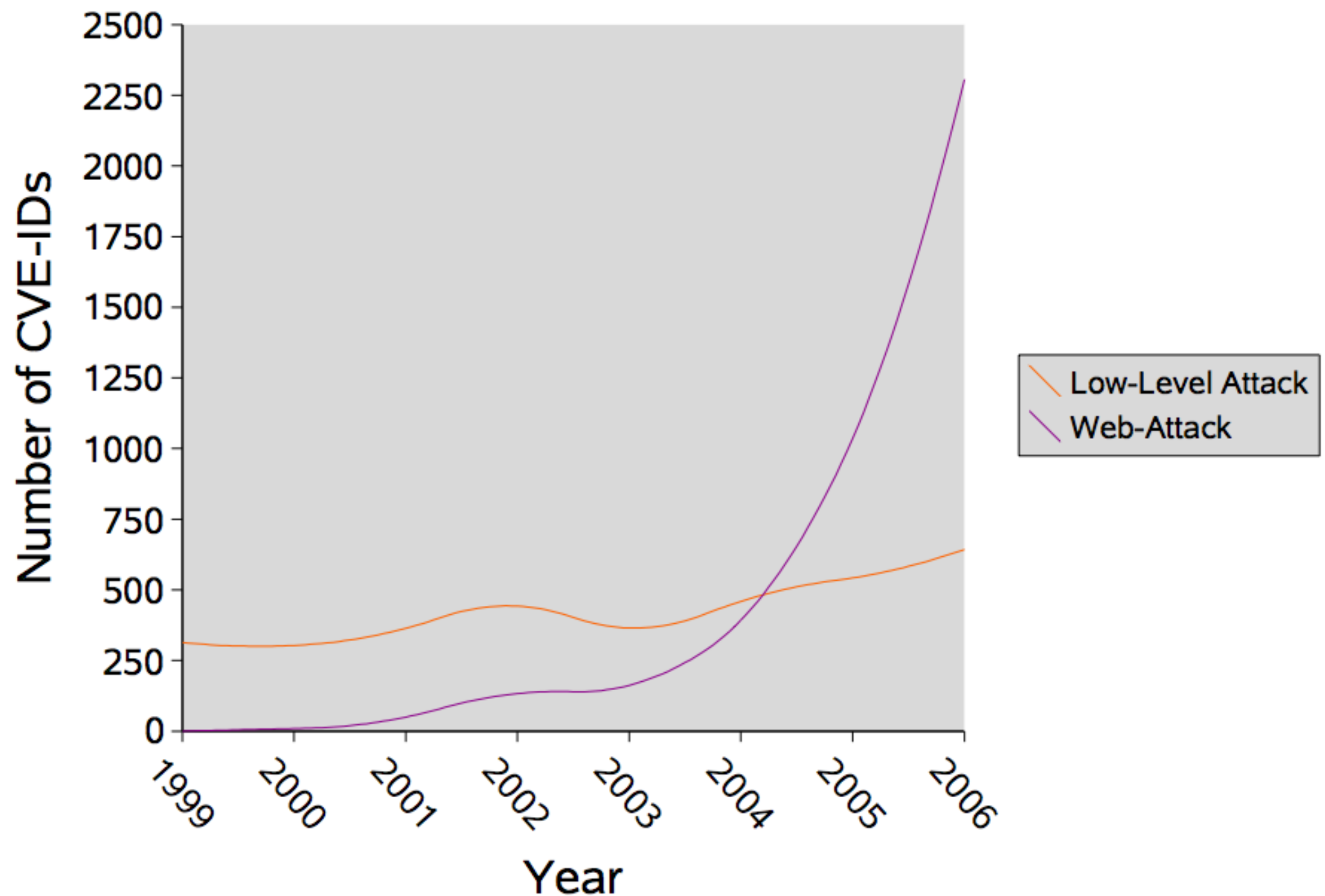


Web-Security Basics

Thomas Biege <thomas@suse.de>

Low-Level and Web-Vulnerabilities



What will you learn here?

- Overview of Common Web-Security Problems
- Authentication in the Web
- Ways to protect your Code

Who and What is not trustworthy?

- *Input*, the Source of all Evil :-o
- *Header*-Fields can be arbitrarily set
- *JavaScript*-Code is running on the Client-Machine and is not under Your Control
- *Cookies*, *hidden* Fields, etc.
- *Output* from Backend-Servers


Input/Output

- LDAP/SQL-Injection
- Command-Execution (*Perl, Shell, PHP, ...*)
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)

The Basic Problems

- Switching of Parser-Context
 - ***Blacklists for Output***: incomplete, easy to use
 - ***Whitelist for Input***: more secure, hard to define?
- check Data-Type, -Length and -Range
- use Naming-Conventions to spot tainted Data more easily (like prefix: `tainted_` and `secure_`)

Stopping SQL-Injections

- use *Prepared Statements*
- *Escaping* to neutralize SQL-Statements that trigger a Change of the Parser-Context (Data  Command)
- Some Languages (like PHP) and Libraries provide Functions for it:
 - *mysql_escape_string()*
 - *mysql_real_escape_string()*

Stopping Shell-Injections

- avoid using *popen()*, *system()*, *\$()*, *`*, *passthru()*, Perl's *open()*, *glob()*, *<>*, etc.
- remove “-” and “--” from the Beginning of every String you pass to Shell-Commands
- *Escaping*, by using:
 - *escapeshellarg()*
 - *escapeshellcmd()*

Stopping XSS

- Do not allow HTML/JavaScript/... in:
 - Everything that can be displayed to other Users
 - direct Input to a Forum
 - Filenames
 - URLs
- *Escaping*, by using:
 - *htmlentities()*
 - *htmlspecialchars()*

Stopping Directory-Traversal and Friends

- work in a “*Chroot*-Environment” (`open_basedir`)
- use random Filenames locally (for Uploads)
- or completely avoid the Filesystem and put File-Contents in a Database
- remove “/” and “\” from the String’s Beginning
- remove “../” and “..\” from the String

Authentication Methods

- Basic Authentication
- Digest Authentication
- Application-based Authentication
 - with Cookies
 - etc.

Without SSL you are
lost!

Basic Auth

- very simple and insecure
- can be configured by Server and/or Application (*PHP*: `header()`)
- Credentials are only base64-encoded
- vulnerable to Replay-Attacks
- vulnerable to Man-in-the-Middle-Attacks
- use it with SSL only

Digest Auth

- Password is NOT sent over the Wire
- NOT vulnerable to Replay-Attacks (only in a limited Time-Window ;-)
- Message-Integrity only optional
- can be configured by Server and Application (*PHP*: `header()`)
- Message-Content is not encrypted (use SSL)
- does NOT protect against Man-in-the-Middle and Downgrading-Attacks

App-based Auth

- ask User for Credentials using a Form
- HTTP is state-less
- Session-Management with Cookies, Fat-URLs, etc.

secure Session- Handling

- Cookie as Container for Authentication-Token
- avoid putting Auth-Token in URL, it can leak via the *Referer-Header* (+ *Session-Fixiation Attacks*)
- *HTTPOnly*- and *Secure*-Flag to avoid leaking
- non-persistent Cookie
- Cookie Lifetime is NOT mandatory!
- re-authenticate before handling security-relevant Client-Requests (like changing Password)

secure Session- Handling

- Session-ID has to be **cryptographically** random
- remove/re-new Session-ID if/on:
 - Logout
 - Session-Context changes (*Session-Fixiation*)
 - Expiration of Lifetime
 - *Referer*-Header shows Redirect from off-site (XSS)
 - User-Agent changes (forgeable) (Cookie stolen)
 - Client-IP changes (unreliable: Proxy) (Cookie stolen)

secure Session- Handling

- **crypto.** MAC to generate/protect Auth-Token:
 - Server-Secret as Key
 - `MAC(username$session-id$timestamp, Key)`
 - Separator (\$) should not be Part of Input (filter)
 - Timestamp is the same as Cookie Exp.-Time
 - Timestamp can also be in a Database
 - use User-ID to lookup Authorisation-Settings, Session-ID, and Timestamp if needed

Information-Leakage

- Forms received over HTTPS, POST over HTTP
- verbose Error-Messages and Comments in JS-Code
- Source-Code Files not handled by Web-Server
- Filename-Suffixes and well-known Files, like:
 - **.bak, *.*~, *.inc, etc.*
 - *WS_FTP, citydesk.html*
 - *etc.*

I want more Details!

- more Presentations:
 - Sanjay Gupta: Security Consideration for Web-Applications
 - Jason Sabin: XSS, Cross Site Scripting, and why would you do that?
 - etc.

Have a Look at:

https://wiki.innerweb.novell.com/index.php/Web_security