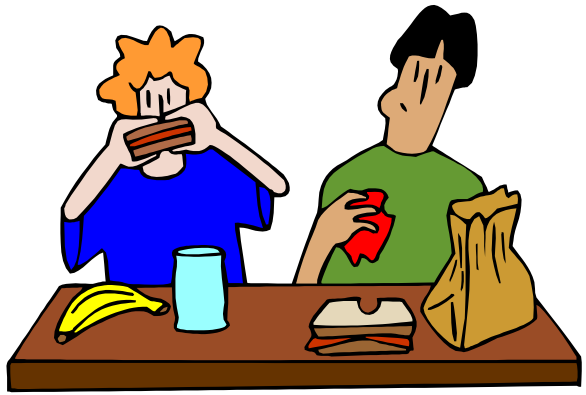
A Guide to Building Secure Web Applications

Part 1: Overview of the OWASP Publication



Brown Bag Presentation by:

Rosie Jergovic, CISSP

October 18, 2002

Outline

- Today's Objectives
- OWASP
- Security Requirements
- Highlights from "A Guide to Building Secure Web Applications"
- Where Do We Go from Here?

Today's Objectives

- Communicate security topics relevant to the development of web applications
- Use OWASP's "Guide to Building Secure Web Applications" as the framework
  - Security Guidelines
  - Architecture
  - Authentication
  - Managing User Sessions
  - Access Control and Authorization
  - Event Logging
  - Data Validation
  - Privacy Considerations
  - Cryptography
- Suggest ways to apply this information

OWASP The Open Web Application Security Project

- An open source reference point for system architects, developers, vendors, consumers and security professionals
- OWASP's goal is to help everyone build more secure web applications and web services
- OWASP will publish guidelines and tools for designing, developing, deploying, and testing the security of web applications and web services
- First publication is: "A Guide to Building Secure Web Applications," Sept. 22, 2002
  - Authors: Mark Curphey, David Endler, William Hau, Steve Taylor, Tim Smith, Alex Russell, Gene McKenna, Richard Parke, Kevin McLaughlin, Nigel Tranter, Amit Klien, Dennis Groves, Izhar By-Gad
- www.owasp.org

Security Requirements

- What is information systems security?
  - Confidentiality - data and services are used only by authorized entities
  - Integrity - data are protected from unauthorized modification in storage, use, and transmission
  - Availability - data, applications, hosts, and networks provide their intended services when they are needed
- What is at stake?
  - Consumer trust in Internet data services
  - Consumer acceptance of mobile commerce
  - Acceptance of our software products
  - Security of network services

Security Requirements

- How much security does a web application require?
  - Zero risk is not practical
  - There are usually multiple ways to mitigate risk
  - Don’t spend $1,000,000.00 to protect $0.10
  - Security is almost always overhead, either in cost or performance

Security Guidelines

- Validate input and output
  - User input and output to and from the system is the route for malicious payloads into or out of the system
  - Allow only explicitly defined characteristics and drop all other data
- Fail securely (closed)
  - Any security mechanism should fail to a state that rejects all subsequent security requests rather than allows them
- Keep it simple
  - Often the most effective security is the simplest security, for end users and administrators
  - If the steps to secure a function or module of the application are too complex, they probably won't be followed properly
  - Complex code is hard to understand, making maintenance error-prone

Security Guidelines

- Use and reuse trusted components
  - When someone else has proven they got it right, take advantage of it
  - Beneficial from both a resource and security perspective
- Only as secure as the weakest link
  - Attackers will find the weakest point and attempt to exploit it
  - Don’t leave all the locks on the front door and leave the back door swinging open
- Security by obscurity won’t work
  - Obscuring information is very different from protecting it
  - It doesn’t work in the long term, and there is no guarantee it will work in the short term

Security Guidelines

- Defense in depth
  - Good systems don't predict the unexpected, but plan for it
  - If one component fails to catch a security event, a second one should catch it
  - Implement a "default deny" stance
- Least privilege
  - The "need to know" approach
  - Systems, applications, functions, modules, etc., should run with the least amount of system privilege needed to do the job
  - Giving the pool man an unlimited bank account to buy chemicals for your pool while you're on vacation is unlikely to be a positive experience!
- Compartmentalization (separation of privileges)
  - Compartmentalizing users, processes, data, and networks helps contain problems if they do occur
  - Give the pool man keys only to the pool house . . .

The Big Picture

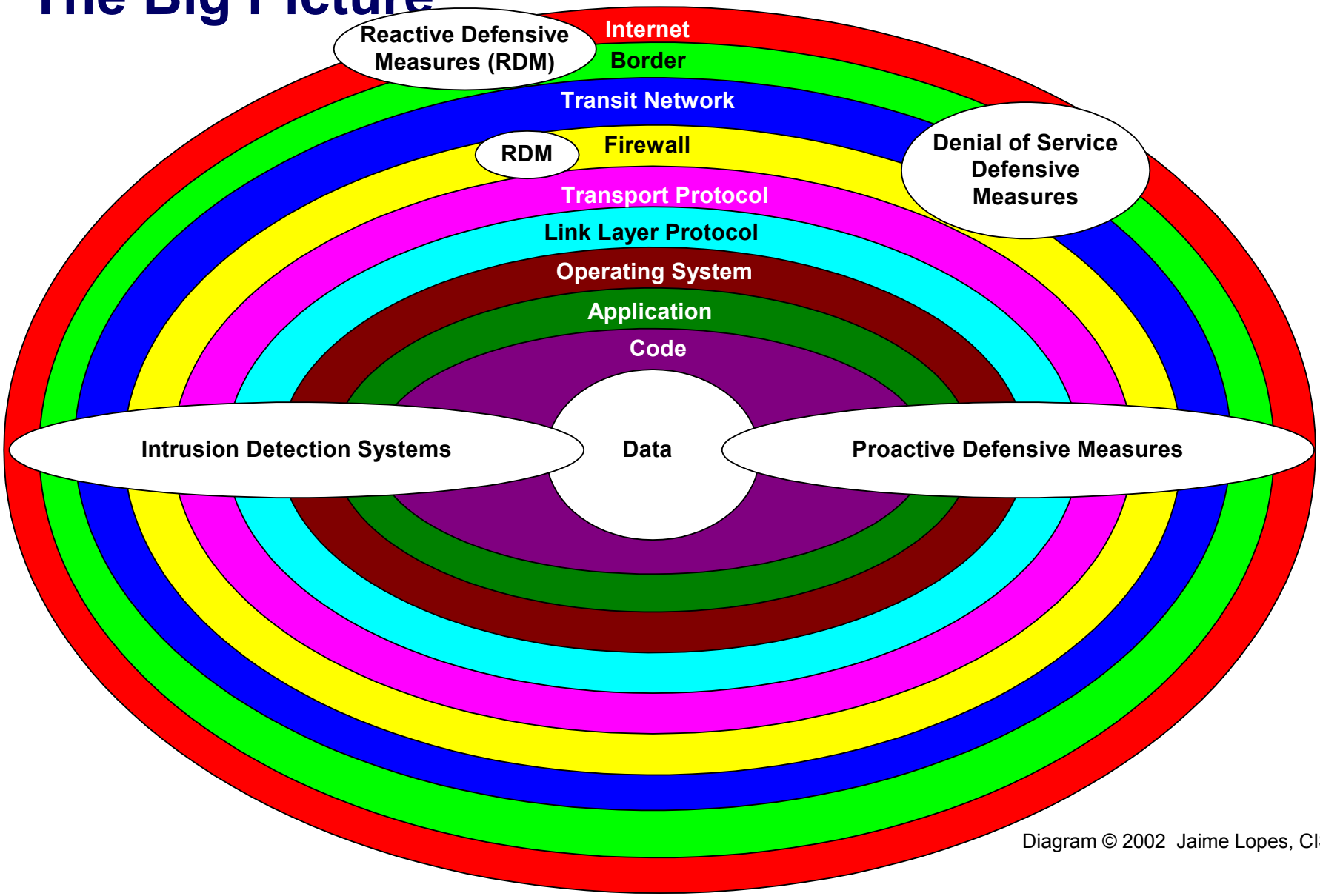
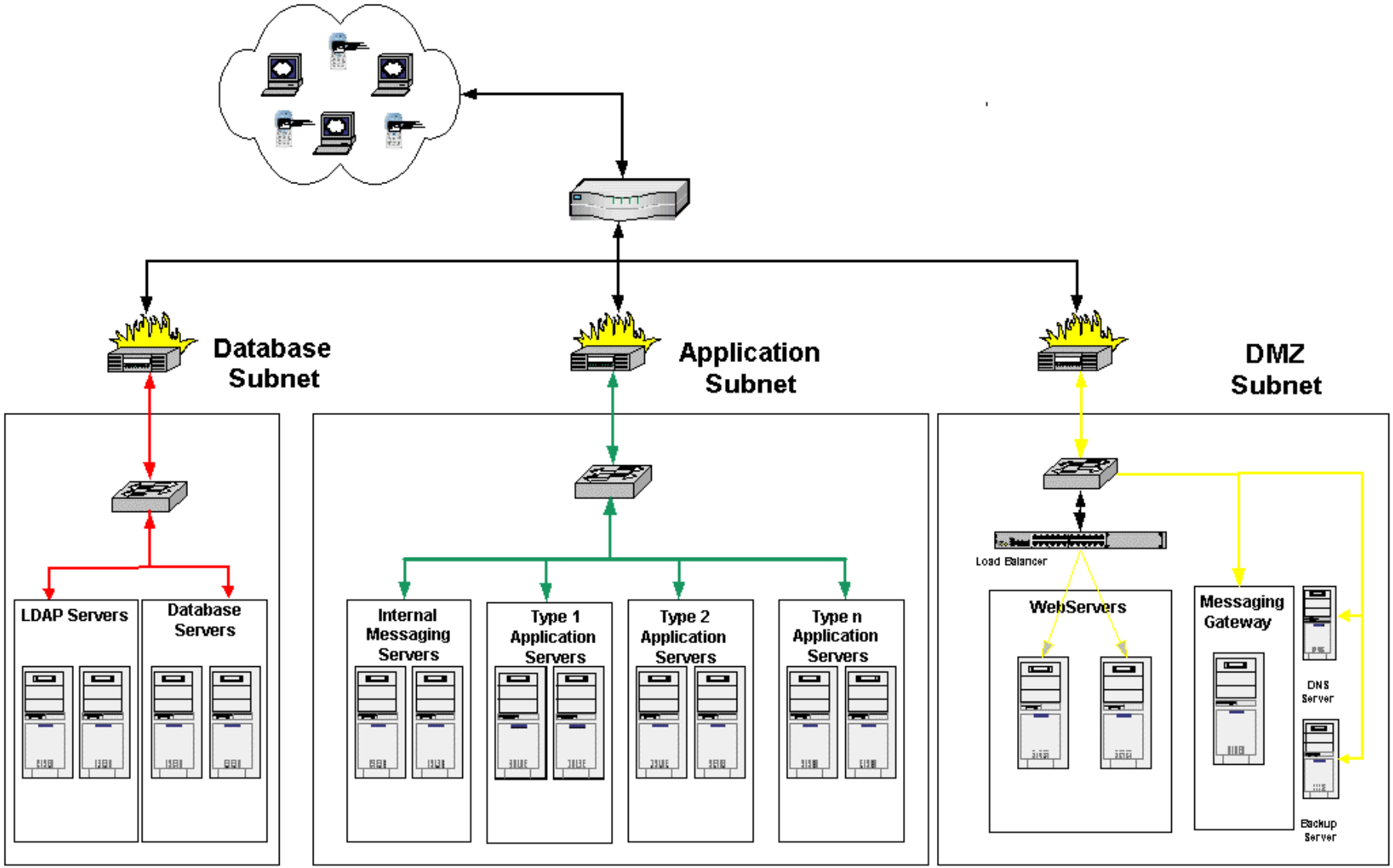


Diagram © 2002 Jaime Lopes, CISSP

Architecture

- Best practice designs separate tiers for
  - content presentation
  - security and control of the user session
  - downstream data storage services
  - administration
- A firewall is not enough!

Architecture



Architecture

- In general, abstract security services from the operating system
  - Too many system compromises have been caused by applications with direct access to parts of the operating system
  - Don't expose OS security interfaces to applications
  - Kernels generally don't protect themselves
    - Exception: trusted operating systems designed with a security kernel, or reference monitor
    - Reference monitor mediates all accesses, is protected from modification, and is verified as correct
  - Contrary to Microsoft's .NET and Sun's JAAS

Authentication

- Authentication – the process of verifying a claimed identity
- Two types
  - User authentication
    - The process of determining that a user is who he/she claims to be
    - Usually happens only once per session
    - May need to re-authenticate during a session, e.g., before a significant monetary transaction
  - Entity authentication
    - The process of determining if an entity is who it claims to be
    - Usually takes place with every request

Authentication

- Consider all input hostile until proven otherwise and code accordingly
  - SSL does not solve the problems of authentication
  - SSL does not protect data once it reaches the client
- HTTP provides protocol-level authentication
  - HTTP Basic (401 status code sent to client) – clear text transmission of username and password
  - HTTP Digest
    - Authentication data obscured by method that includes MD5 hashing before transmission
    - Protocol includes additional protection for replay attacks, mutual authentication, and integrity

Authentication

- Form-based authentication
  - Essential for authentication forms to be submitted using a POST request
    - GET requests show up in the user's browser history and may be visible to other users of the same browser
  - Use SSL to protect POST request in transmission
  - Never prefill password fields for a user
  - Best practice
    - Have a blank password field asking the user to confirm the current password
    - Use two password fields to enter and confirm a new password
    - Keep the ability to change a password on a page separate from that for changing other profile information
- Highest authentication security requires client side authentication, likely using Public Key Infrastructure (PKI)
  - Emerging standards for XML-based key management: www.w3.org.TR/xkms

Authentication

- Entity authentication
  - Cookies
  - Do not use the Referer Header – it is implemented by the user's browser and can be changed at will
- Infrastructure authentication
  - Inherent insecurities of DNS make the use of IP Address or DNS names an unreliable source of authentication data
  - IP address spoofing also makes use of this data for authentication unreliable.
  - Consider using X.509 certificates or implementing SSL

Authentication

- Best practices for usernames and passwords
  - Few security requirements for usernames, but consider privacy issues (e.g., real names, SSN, tax IDs, MSISDN …)
  - Encrypt passwords before storage, e.g., using a hash algorithm
  - Ensure password strength, e.g., at least 8 characters, one alphanumeric, one mixed case, at least one special character
  - Employ password lockout mechanisms
  - Age passwords and limit reuse of passwords
  - If a system distributes new passwords (manually or programmatically), the password should be set to change the first time the new user logs on with the changed password

Authentication

- Best practices for usernames and passwords
  - Single Sign On (SSO) across multiple DNS domains
    - Microsoft Passport and Project Liberty
    - SSL-based protection schemes are subject to man-in-the-middle attacks
    - To do secure SSO, the token must be protected outside of SSL (“out of band”)
    - Use symmetric key algorithms with a pre-exchanged key and including a timestamp in the token to prevent replay attacks

Managing User Sessions

- Cookies
  - Persistent and Secure
    - Stored in a text file on the client and valid until expiry date
    - Requires SSL protection during transmission
  - Persistent and Non-secure
    - Stored in a text file on the client and valid until expiry date
    - Protection by SSL during transmission is optional
  - Non-persistent and Secure
    - Stored in RAM on the client and destroyed when the browser is closed or the cookie is explicitly killed by a log-off script
    - Requires SSL protection during transmission
  - Non-persistent and Non-secure
    - Stored in RAM on the client and destroyed when the browser is closed or the cookie is explicitly killed by a log-off script
    - Protection by SSL during transmission is optional

Managing User Sessions

- Session tokens
  - Cryptographic algorithms
    - All session tokens should be user unique, non-predictable, and resistant to reverse engineering
    - Use a trusted random number generator
    - Map session tokens in some way to a specific HTTP client instance to prevent hijacking and replay attacks
      - Example: use page tokens which are unique for any generated page and may be tied to session tokens on the server
    - Do not base a session token algorithm on or use as variables any user personal information
  - Appropriate key space
    - A token's cryptographic key space should be sufficiently large enough to prevent brute force attacks

Managing User Sessions

- Session management schemes
  - Expire session tokens on the HTTP server when the session ends
    - Static session tokens are vulnerable to capture and brute-force cracking if they are stored indefinitely on the client or logged/cached in proxy servers
  - Regenerate session tokens while the session is active
    - Results in a smaller window of time for replay exploitation of each legitimate token
    - Can be based on number of requests or time
  - Build methods to detect session forging/brute-forcing and/or lockout
    - Use “booby trapped” session tokens that never actually get assigned but will detect if an attacker is trying to brute force a range of tokens
    - Design anomaly/misuse detection hooks to detect if a legitimate user tries to manipulate their token to gain elevated privileges

Managing User Sessions

- Session management schemes
  - Re-authenticate users before significant actions
    - Sensitive user actions (e.g., money transfer) should require the user to re-authenticate or be reissued another session token immediately prior to significant actions
    - Segment data and user actions to the extent where re-authentication is required upon crossing certain boundaries to prevent some types of cross-site scripting attacks that exploit user accounts
  - Encrypt session tokens during transmission
    - Use web encryption technologies (e.g., SSL or TLS protocols) to safeguard the session token from replay or hijacking attacks if it is captured in transit through network interception

Managing User Sessions

- Session management schemes
  - Use page-specific tokens in conjunction with session-specific tokens
    - When used with encrypted transport, helps to ensure that the client on the other end of the session is the same client which requested the last page in a given session
    - May be stored in cookies or query strings and must be completely random
    - Can avoid sending session token information to the client by using page tokens and creating a mapping between them on the server side
  - Overwrite session tokens on logout
    - Internet kiosks and shared internet environments usually maintain the same browser thread; the browser only destroys session cookies when the browser thread is torn down
    - Overwrite session cookies when the user logs out of the application

Access Control and Authorization

- Limits what users can do, which resources they have access to, and what functions they are allowed to perform on the data
- Authorization
  - The act of checking to see if a user has the proper permission to access a particular file or perform a particular action
- Access Control
  - The more general way of controlling access to web resources, including restrictions based on things like the time of day, the IP address or domain of the client browser, the type of encryption the HTTP client can support, number of times the user has authenticated that day, etc.
  - Any access control mechanism depends on effective and forge-resistant authentication controls for authorization

Access Control and Authorization

- Discretionary access control
  - Restricts access to information based on the identity of users and/or membership in certain groups
  - Decentralized model - the owner of the controlled resource can change its permissions at will
- Mandatory access control
  - Secures information by assigning sensitivity labels on information and comparing this to the level of sensitivity at which a user operates
  - Appropriate for extremely secure systems, such as, military applications or mission critical data applications

Access Control and Authorization

- Role-based access control
  - Access decisions are based on an individual's roles and responsibilities within the organization or user base
  - Centralized administration
  - Scalable

Event Logging

- Importance of logging
  - Provides key security information about a web application and its associated process and integrated technologies
  - Makes individual users accountable for their actions
  - Often the only record of suspicious behavior; can feed IDS
    - Permits reconstruction of events
    - May be needed in legal proceedings to prove wrongdoing (actual handling of log data is crucial)
- What to log
  - In general, time of event, initiating process or owner of process, detailed description of the event
  - Reading/writing/modification/deletion of data
  - Network communications at all points
  - All authentication/authorization events
  - All administrative functions, regardless of overlap
  - Miscellaneous debugging information that can be enabled/disabled on the fly

Event Logging

- Best practices for log management
  - Collect and consolidate logs on a separate dedicated logging host
  - Encrypt network connections and log data contents to protect confidentiality and integrity
  - Set log file attributes so that only new information can be written (older records cannot be rewritten or deleted)
  - Copy logs at regular intervals, depending on size and volume
  - Verify regularly that logging is operational
  - Copy log files to permanent storage and include in data center's backup strategy
  - Dispose of log files according to company record retention policies
  - Synchronize all logging components with a time server so that all logging can be consolidated effectively without latency errors
    - Harden the time server and limit its services to time

Data Validation

- One of the most important aspects of designing a secure web application
- Applies to input to and output from a web application
- Validation strategies
  - Accept only known valid data
    - Accept only input that is known to be safe and expected
    - The best possible strategy, but not always feasible
  - Reject known bad data
    - Can limit exposure
    - Relies on the application knowing about specific malicious payloads
    - Easy for knowledge base of web application attack signatures to be out-of-date
  - Sanitize all data
    - An effective second line of defense
    - Too difficult to be relied upon as a primary defense

Data Validation

- All three data validation strategies must check
  - Data type (extremely important – don't let an object through if you are expecting a string!)
  - Syntax
  - Length
- Never rely on client-side data validation
  - Data validation must be done on the trusted server or under the control of the application
  - Client-side data validation can always be bypassed
    - An attacker can watch the return value and modify it at will

Privacy Considerations

- Warn users about the dangers of communal web browsers
  - Pages may be retained in the browser cache
  - Recommend logging out and closing the browser to kill session cookies
  - Temp files may remain
  - Proxy servers and other LAN users may intercept traffic
- Design sites with the assumption that no part of a client is secure and make no assumptions about integrity

Privacy Considerations

- Display personal data only when absolutely needed
  - Set pages to pre-expire
  - Set the no-cache meta tags
  - Set the no-pragma-cache meta tags
  - Otherwise, mask personal data, displaying only a subset
- Offer an enhanced privacy login option to the user, which
  - Sets pages to pre-expire
  - Sets the no-cache meta tags
  - Sets the no-pragma-cache meta tags
  - Uses SSL or TLS
- Keep sensitive data out of the browser history by using POST for all form submissions

Cryptography

- Important part of building secure web applications
- No silver bullet!
- Beware of snake oil cryptography
  - www.interhack.net/people/cmcurtin/snake-oil-faq.html
- Good cryptography is based on the secrecy of the key, not the security of the algorithm
  - 64-bit RC5 key broken in 4 years, 10 months of computing …
- A cryptographic system typically consists of algorithms, keys, and key management facilities
  - Symmetric cryptographic systems (one shared key)
  - Asymmetric cryptographic systems (two keys; one public and one private)

Cryptography

- Symmetric cryptography
  - Single private key to encrypt and decrypt data
  - Algorithms are fast and suitable for processing large streams of data
  - Presumes two parties have agreed on a key and have been able to exchange that key in a secure manner before communicating.
  - Usually mixed with public key algorithms to obtain a blend of security and speed

Cryptography

- Asymmetric, or public key, cryptography
  - Private key that must be kept from unauthorized users and a public key that can be known by anyone
  - Public and private key are mathematically linked
    - Data encrypted with the public key can be decrypted only by the private key
    - Data signed with the private key can only be verified with the public key
  - Useful for small amounts of data
    - Digital signature (nonrepudiation, authentication)
    - Securing transmission of shared keys

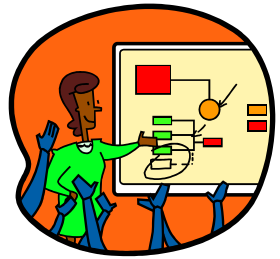
Cryptography

- Hash value
  - One-way mathematical algorithms that take an arbitrary length input and produce a fixed length output string – the hash value
  - A unique and extremely compact numerical representation of a piece of data (e.g., MD5 produces a 128- bit value)
  - Computationally improbable to find two distinct inputs that hash to the same value
  - Useful
    - Digital signatures
    - Integrity protection
    - Allowing a party to prove they know something without revealing what it is, e.g., password protection schemes

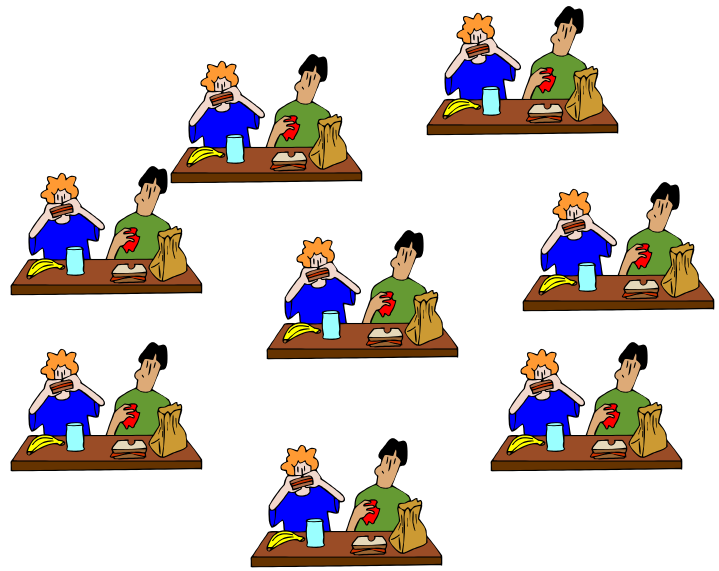
Cryptography

- Implementing cryptography
  - Cryptographic toolkits and libraries
    - Java Cryptography Extensions (JCE-cannot export) and Java Secure Socket Extensions (JSSE) (part of JDK 1.4)
    - Cryptix
      - Open source clean-room implementation of the official JCE 1.2 API as published by Sun
      - PGP library
    - OpenSSL
    - Legion of the Bouncy Castle – Java cryptography library for JSSE and J2ME
  - Key generation – design securely
  - Random number generation
    - Use for generating keys so it is infeasible to reproduce or predict them
    - EGADS (www.securesoftware.com/egads.php); YARROW (www.counterpane.com/yarrow.html)
  - Key lengths – large enough to provide “cover time”

Where do we go from here?



- Security-focused architecture workshops
  - Use the OWASP guide as a template during architecture workshops to ensure that security concerns are considered and informed tradeoffs are made
- Future topics for security brown bags
  - Preventing common problems
    - Generic meta-characters problem
      - Attacks on the users
      - Attacks on the system
      - Canonicalization
    - Parameter manipulations
    - Miscellaneous
  - Secure coding practices



Where do we go from here?

- Version 2 of the Guide is due January 2003
  - Topics being considered include
    - Language security
    - Java
    - J2E
    - Project Liberty
    - SAML
    - Error handling
- OWASP Testing Framework Group
  - Working on a comprehensive web application testing methodology
    - Will cover “white box” (source code) analysis
    - Will cover “black box” (penetration test) analysis
  - Expected late 2002

Security is a Bodyguard, not a Policeman