CS 465

24 DEADLY SINS

Slides by Kent Seamons and Tim van der Horst
Last Updated: Dec 2, 2013

Implementation Sins

Sin 11 – Failing to Handle Errors

- When a programmer fails to handle an error, the program could get into an insecure state or crash
- Program termination opens up a denial of service vulnerability
- Revealing too much information about an error can aid an attacker
- Sample code that is copied and pasted often leaves out error handling

Sin 11 – Failing to Handle Errors

- Five variants
  - Yielding too much information
  - Ignoring errors
  - Misinterpreting errors
  - Using useless return values
  - Using non-error return values
- Redemptive Steps
  - Only real step is to make sure you check return values when appropriate

Sin 11 – Failing to Handle Errors

- Do check the return value of every security-related function
- Do check the return value of every function that changes a user setting or a machine-wide setting
- Do make every attempt to recover from error conditions gracefully, to help avoid DOS problems
- Consider using code annotations if they are available, for example in Microsoft Visual C++
- Do not rely on error checking solely using assert
- Do not leak error information to untrusted users

Sin 12 – Information Leakage

- An attacker obtains data that leads to a security breech
  - Accidental, Intentional, Mistake
- Examples
  - Sides Channels
    - Timing Channels
    - Storage Channels (file names, file sizes)
  - Too much information
    - Detailed version information
    - Host network information
    - Application information
    - Path information
    - Stack layout information
- Along with checking untrusted input, there is a need to review output to untrusted users

Sin 12 – Information Leakage

- Do define who should have access to what error and status information
- Do identify all the sensitive or private data in your application
- Do use appropriate operating system defenses such as ACLs and permissions
- Do use cryptographic means to protect sensitive data
- Do not disclose system status info to untrusted users
- Consider using other operating system defenses such as file-based encryption

Sin 13 – Race Conditions

- When two execution contexts (threads or processes) interfere with one another
- Usually a failure to handle concurrency correctly
- The file changes between the time it was checked for valid permissions and the time an operation occurs (delete)
  - TOCTOU – Time of check, time of use

Sin 13 – Race Conditions

- Do write code that doesn't depend on side effects
- Do be very careful when writing signal handlers
- Do not modify global resources without locking
- Consider writing temporary files into a per-user store instead of a world-writable space

Sin 14 – Poor Usability

- Security is (almost) never the user's priority
  - Example – Vista User Account Control (UAC)
- Security only works if the secure way happens to be the easy way – Scott Culp
- Presenting security information to users
  - Too little appropriate information
  - Too much information
  - Too many messages
  - Inaccurate or generic information
  - Errors with only error codes

Sin 14 – Poor Usability

- Example Sins
  - TLS Certificate Authentication
  - Root Certificate Installation
- Redemption Steps
  - Make the UI simple and clear
  - Make security decisions for users
  - Make selective relaxation of security policy easy
  - Clearly indicate consequences
  - Make it actionable
  - Provide central management – OS level rather than application by application

Sin 14 – Poor Usability

- Do understand your users' security needs, and provide the appropriate information to help them get their jobs done
- Do realize that just because you understand some security text, that does not mean your users do
- Do default to a secure configuration whenever possible
- Do provide a simple, and easy to understand, message, and allow for progressive disclosure if needed by more sophisticated users or admins
- Do make security prompts actionable
- Do not dump geek-speak in a big honking dialog box
  - No user will read it
- Do not make it easy for users to shoot themselves in the foot
  - Hide options that can be dangerous
- Consider providing ways to relax security policy selectively, but be explicit and clear about what the user is choosing to allow

Sin 15 – Not Updating Easily

- This sin covers a lot of ground
- Making patches difficult to install
- Getting your users hacked when they update their software
- Make user install additional unwanted software
- Prompt fatigue
- Update without notifying
- Forcing reboot
- Trusting DNS

Sin 15 – Not Updating Easily

- Do sign any code or data your download onto a user's system
- Do validate the signature correctly
- Do write temporary files to a trusted location, not a shared temporary folder
- Do write your binary data to a secure location
- Do make your patches easy to install. If your app will be deployed widely in an enterprise, make sure patches can be installed across many systems easily.
- Do write patches into a secured area
- Do not trust the network
- Do not trust DNS
- Do not write temporary files to a shared temporary folder

Sin 16 – Executing Code with Too Much Privilege

- Do plan for least privilege early in your development cycle
- Do run your code with the lowest possible privilege
- Do not run your code with administrative or root capabilities simply because “stuff works”
- Consider dropping unneeded privileges as soon as possible to reduce exposure
- Consider Linux and BSD capabilities

Sin 17 – Failure to Protect Stored Data

- Do apply appropriate permissions or ACLs to files
- Do analyze all ACLs and permissions you set
- Do encrypt files that store sensitive data
- Do store encryption data using operating system primitives where possible
- Do install binaries to protected locations in the file system

Sin 17 – Failure to Protect Stored Data

- Do scan the file system, pre/post installation of your product, to detect weak ACLs or permissions
- Do not create weak ACLs, such as Everyone: Full Control or weak permissions such as World:Write
- Consider using permissions and encryption together
- Consider adding an integrity defense to the sensitive data such as an HMAC or signature

Sin 18 – The Sins of Mobile Code

- Do write mobile code in safer technologies such as .NET and Java
- Do assume your mobile code container will render *malicious* mobile code
- Do fuzz-test your mobile code methods and properties
- Do use as many constraining defenses as possible in your mobile code container
- Do digitally sign your mobile code with a code-signing private key and certificate
- Do SiteLock ActiveX controls
- Do not leak sensitive data from mobile code

Cryptographic Sins

Sin 19 – Use of Weak Password-Based Systems

- Password compromise
- Allowing weak passwords
- Iterated passwords (cougars1,cougars2)
- Never changing a password
- Default passwords
- Replay attacks
- Brute-force attacks against password verifiers

Sin 19 – Use of Weak Password-Based Systems

- Storing passwords instead of password verifiers
- Online attacks, including allowing these to create a denial of service attack
- Revealing whether a failure is due to an incorrect username or password
- Returning a forgotten password instead of resetting it

Sin 19 – Use of Weak Password-Based Systems

- Examples
  - MAC OS email client sent email password in the clear before the user specifies that SSL/TLS should be used
  - TENEX bug that leaked timing information
  - Paris Hilton Hijacking
    - Attacker reset her password by answering her “secure” question – what is the name of your pet?
  - Sarah Palin Yahoo Email Compromise
    - Answering questions to reset password

Sin 19 – Use of Weak Password-Based Systems

- Do ensure passwords are not sent in the clear
- Do give a single error message for failed login attempts
- Do log failed password attempts
- Do use strong, salted cryptographic one-way function based on a hash for password storage
- Do provide a secure mechanism for people to change passwords
- Do not make it easy for customer support to reset a password over the phone
- Do not ship with default accounts and passwords
- Do not store plaintext passwords on the server
- Do not store passwords in code
- Do not log the failed password
- Do not allow short passwords

Sin 19 – Use of Weak Password-Based Systems

- Consider:
  - Storage algorithm PBKDF2 that supports making the one-way hash computationally expensive
  - Multifactor authentication
  - Zero-knowledge password protocols
  - One-time password protocols
  - Ensuring passwords are strong programmatically
  - Recommending strategies for coming up with strong passwords
  - Automated ways to reset passwords

Sin 20 – Weak Random Numbers

- PRNG vs. CRNG
- Popular languages have weak PRNG – see the table in your book

Sin 20 – Weak Random Numbers

- Do use the system CRNG
- Do make sure the CRNG is seeded with at least 64 bits of entropy, preferably 128 bits
- Do fail the user's current operation if the CRNG fails for any reason
- Do not use a non-cryptographic PRNG for a cryptographic operation
- Do not fall back to a PRNG if the CRNG fails
- Consider using hardware RNG in high-assurance situations

Sin 21 – Using the Wrong Cryptography

- Using home-grown cryptography
- Using a weak cryptographic primitive
- Using the wrong primitive
- Failing to use a salt
- Not providing an integrity check
- Key re-use
- Verifying a hash value improperly

Sin 21 – Using the Wrong Cryptography

- Do use SSL3 or TLS 1 for channel protection
- Do use random salt when appropriate
- Do use a random IV for chained block cipher
- Do use appropriate cryptographic algorithms (AES, SHA-2)
- Do not build your own crypto
- Do not hash concatenated data
- Do not build your own secure protocol when a higher-level protocol will work just as well
- Do not use MD4 or MD5, DES, RC4, ECB
- Do not use SHA-1 in new code

Networking Sins

Sin 22 – Failing to Protect Network Traffic

- Network attacks take a variety of forms
  - Eavesdropping
  - Replay
  - Spoofing
  - Tampering
  - Hijacking

Sin 22 – Failing to Protect Network Traffic

- Do use a strong initial authentication scheme
- Do perform ongoing message authentication
- Do encrypt all data that is sensitive
- Do use TLS for your on-the-wire protocols
- Do not hardcode keys
- Do not hesitate to encrypt data for efficiency reasons
- Do not ignore the security of your data on the wire
- Consider using network-level technologies to further reduce exposure – firewalls, VPNs, and load balancers.

Sin 23 – Improper Use of PKI, especially SSL/TLS

- Do understand what services you require from SSL
- Do understand what your SSL libraries check by default
- Do verify the certificate
  - integrity, ownership, expiration, revocation, usage
- Do not continue authentication if the certificate validation fails for any reason
- Do not only check the name in a certificate – anyone can place any name in a certificate
- Consider using an OCSP responder when validating certificates in a trusted chain to ensure that the certificate hasn't been revoked

Sin 24 – Trusting Network Name Resolution

- DNS is not secure
- The problem is language-independent
- UDP poses larger threat than TCP
- DNSSEC is one solution that DHS is promoting (adds authentication/integrity)

Sin 24 – Trusting Network Naming Resolution

- Do use cryptography to establish the identity of your clients and servers. A cheap way to do this is through TLS. Be sure to completely validate certificates.
- Do not trust DNS information - it isn't reliable!
- Consider specifying IPSec for the systems your application will run on