Top 5 Security Issues Developers Don't Know About

Neelay S. Shah
Senior Security Consultant
Foundstone Inc. , A Division of McAfee
neelay.shah@foundstone.com

OWASP
March 13, 2008
NY/NJ Chapter

The OWASP Foundation
http://www.owasp.org

Goal

- Talk about the security issues that are frequently seen in the wild and yet the awareness regarding the same in the developer community is very low

Agenda

- Top 5 Security Issues Developers Don't Know About
  - Inter-process communication
  - Launching New Processes
  - Using Cryptography While Updating Applications
  - ActiveX Controls
  - Integrating and Leveraging Third Party Controls
- Questions

Approach

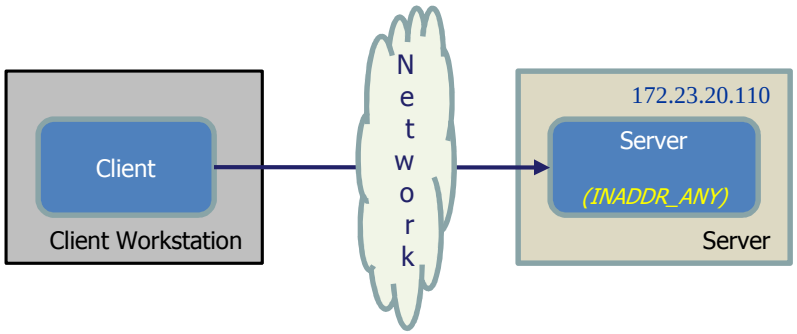
- Security Issue
  - Discuss the issue
  - Impact
  - Exploitation/Mitigations
  - Recommendations
  - Identifying the security issue

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Sockets
  - Background
    - Allows for bi-directional communication between the communicating processes
    - Supported by almost every operating system and hence is the most widely used communication instrument for client server applications
  - Socket Hijacking
    - Discussion
      - It is possible to bind more than one socket to the same port
      - When a client connects, the underlying socket library decides which socket gets the data by looking at the sockets and forwards the data to the socket whose binding is more specific
      - Attacker can hijack the listening socket

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Typical setup for client-server application communicating using sockets



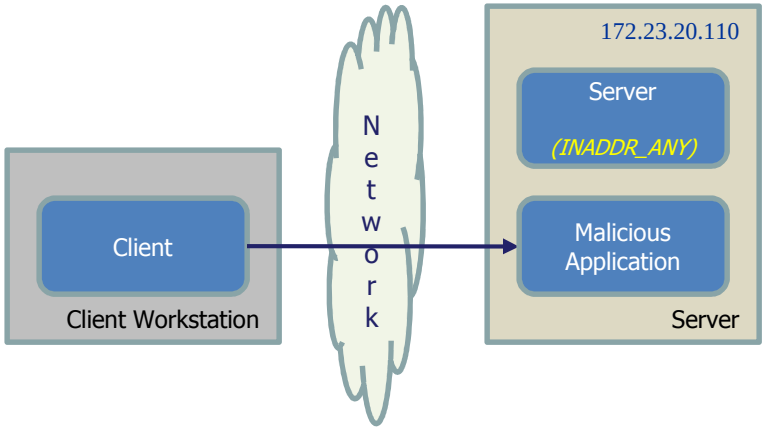
1. Server application binds the listening socket to INADDR_ANY and is waiting for incoming connections

2. User launches the client application

3. Client application uses the server's IP address to connect to the server application.

4. Client application successfully connects to the server and continues to function.

- Socket Hijacking



1. Server application binds the listening socket to INADDR_ANY and is waiting for incoming connections

2. Malicious application binds the listening socket to the **specific IP address** and is also now waiting for incoming connections

3. User launches the client application

4. Client application uses the server's IP address to connect to the server application.

5. Malicious server application gets the client application connection and has successfully hijacked the real server socket.

6. Client application continues to function thinking it has connected to the real server application

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Sockets
  - Socket Hijacking
    - Impact
      - Set up a spoof server
        - » Man-In-The-Middle attacks
        - » Information Disclosure attacks
        - » Denial of service attacks
    - Exploitation / Mitigations
      - Need access to the system
      - Only impacts Windows in out-of-the box configuration
      - On Windows XP SP2 and later, administrative privileges are needed to bind to ports 0 - 1023

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

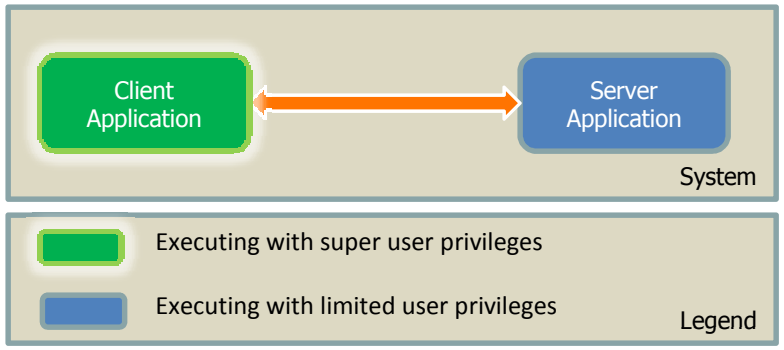
- Sockets
  - Socket Hijacking
    - Recommendation
      - As far as possible while opening a listener socket, bind the socket to a specific IP address
      - Use SO_EXCLUSEADDRUSE option to ensure that only one socket can be bound to the same port at a time
- Identifying socket hijacking issues
  - Typical applications which are likely to be vulnerable to socket hijacking
    - Client – Server applications communicating using sockets
  - Relevant API's to look out for:
    - bind()
    - setsockopt(); option: SO_REUSEADDR

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Named Pipes
  - Background
    - Allows for bi-directional communication between the communicating processes
    - Can be used for communicating between processes executing on different machines but typically are used for communication between processes on the same machine
  - Insecure Named Pipe Creation/Connection
    - Discussion
      - Allows the server end to identify / impersonate the client end of the named pipe
      - Typically used to by the server process to ensure that the client requests are processed with appropriate privileges

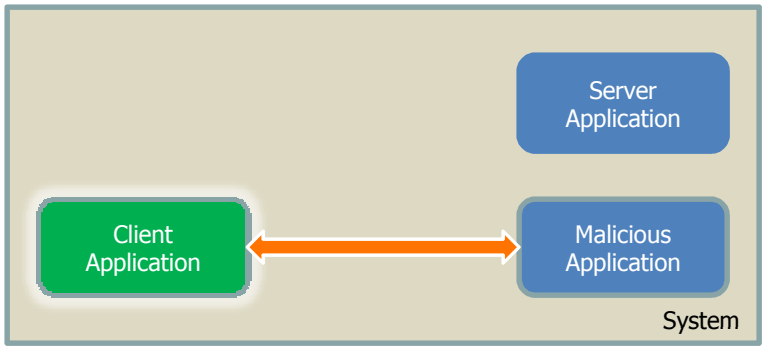
Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Typical setup for processes communicating using named pipes



1. Server application starts listening and is waiting for incoming connections on the **named pipe**
2. User launches the client application
3. Client application connects to the server using named pipe
4. Client application successfully connects to the server and continues to function.

- Named Pipes Security Issue



1. Malicious application creates and starts listening on the **named pipe**
2. Server application starts and attempts to create the named pipe and starts listening on the named pipe
3. Server application ends up creating only a new instance of the existing named pipe
4. User launches the client application
5. Client application connects to the server using named pipe
6. Malicious server application gets the client application connection and has successfully hijacked the real server connection
7. Client application continues to function thinking it has connected to the real server application

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Named Pipes
  - Insecure Named Pipe Creation/Connection
    - Impact
      - Set up a spoof server
        - » Privilege Escalation
        - » Man-In-The-Middle attacks
        - » Information Disclosure attacks
        - » Denial of service attacks
      - Insecure Configuration
        - » If the named pipe already exists, the OS only creates a new instance of the same named pipe
        - » If the named pipe already exists, the OS does not apply the security descriptor (which controls the permissions on the named pipe for e.g. who can read from the named pipe, who can write to the named pipe etc.

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Named Pipes
  - Insecure Named Pipe Creation/Connection
    - Exploitation / Mitigations
      - Need access to the system
      - Need to be able to run the malicious code before the legitimate vulnerable code runs
      - On Windows 2004 SP4+, Windows XP SP2+, Windows Server 2003 and later, the "SeImpersonatePrivilege" privilege is needed to impersonate the client end of the named pipe
    - Recommendation
      - Create the named pipe securely
        - » Ensure that the named pipe does not already exist
        - » Apply strong ACL on the named pipe
      - Connect to a named pipe securely
        - » If the server does not need to impersonate the client, ensure that the option that allows the server end of the named pipe to impersonate the client is turned off

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Identifying "Named Pipes" security issues
  - Typical applications which are likely to be vulnerable
    - Applications that comprise of client-server processes (executing at different privilege levels) and the server processes client requests with only the appropriate privileges
  - Relevant API's to lookout for
    - CreateNamedPipe()
    - CreateFile()
    - ImpersonateNamedPipeClient()

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Mutexes / Events
  - Background
    - Mutexes and Events are typically used for interprocess synchronization for e.g. to signal the occurrence of an particular incident or to guard access to a shared resource such as the database
  - Insecure Mutex / Event Creation
    - Mutexes / Events are "securable" kernel objects and as such an appropriate security descriptor must be used while creating mutexes / events
    - Impact
      - Insecure Configuration
        - » If the kernel object already exists, the OS only returns an handle to the already existing one
        - » If the kernel object already exists, the OS does not apply the security descriptor (which controls the permissions on the object for e.g. who can lock/signal/read from the object, who can unlock/reset/write to the object etc.

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Mutexes / Events
  - Insecure Mutex / Event Creation
    - Impact
      - Denial of Service Attacks
        - » If the application waits "infinitely" for the kernel object to be released before proceeding, then an attacker could successfully launch a denial of service attack
      - Privilege Escalation
        - » If a low privileged process manages to change the state of the mutex / event it could result in a privilege escalation depending on how the mutex / event is being used
    - Exploitation / Mitigations
      - Need access to the system
    - Recommendations
      - Create the mutex / event securely
        - » Ensure that the mutex / event does not already exist
        - » Apply strong ACL on the mutex / event

Security Issues Involved With Using Interprocess Communication/Synchronization Primitives

- Identifying "Mutexes / Events" security issues
  - Typical applications which are likely to be vulnerable
    - Applications that comprise of a worker process and a monitor process. The sole job of the monitor process is to start the worker process if it has stopped for e.g. the worker process crashes unexpectedly and / or restart the worker process when needed for e.g. after installing updates
    - Applications that comprise of multiple processes that need to synchronize their execution with each other
  - Relevant API's
    - CreateMutex()
    - CreateEvent()
    - WaitForSingleObject() / WaitForMultipleObjects()

Security Issues Involved With Creating New Processes

- Insecure Process Creation
  - Discussion
    - Frequently as part of the functionality, applications need to be able to launch (trusted / untrusted) external processes
  - Impact
    - Privilege Escalation
      - If the application is a Windows service or an Unix daemon, it will typically execute with higher privileges and as such the new process will also launch with higher privileges.
        - » If the attacker controls which process gets executed he / she can successfully exploit the vulnerability to compromise the application / system
        - » If the application only allows "trusted" processes to be launched, the attacker still can cause privilege escalation if the application does not launch the new process securely
  - Exploitation / Mitigations
    - Need Local Access

Security Issues Involved With Creating New Processes

- Insecure Process Creation
  - Recommendations
    - Launch new process with only the appropriate privileges
    - Launch new process securely
- Identifying "Insecure Process Creation" security issue:
  - Typical applications which are likely to be vulnerable
    - Applications that expose a "Scheduled Tasks" feature
    - Applications that execute when the user logs on
  - Relevant API's:
    - CreateProcess()
    - CreateProcessAsUser()

Security Issues Involved With Using Cryptography Using Cryptography While Updating Applications

- Insecure Integrity Checking
  - Background
    - Frequently applications expose "Check For Update / Auto Update" functionality, wherein the application downloads updates from the server and installs them locally
  - Discussion
    - No integrity checking
    - Weak integrity checking algorithms used for e.g. MD4
    - Hash / Public key used to verify the integrity is downloaded in clear without any integrity checking mechanism for the same
  - Impact
    - Man-In-The-Middle attacks
    - Privilege Escalation
    - System Compromise
    - Denial of Service
  - Exploitation / Mitigations
    - Remotely Exploitable

Security Issues Involved With Using Cryptography Using Cryptography While Updating Applications

- Insecure Integrity Checking
  - Recommendations
    - Verify the downloaded update to ensure that it has not been tampered in the transit and it has been published by the intended server before installing the update
    - Ensure that the hash / public key used to verify the integrity of the update is exchanged securely

- Identifying cryptography security issues
  - Typical applications which are likely to be vulnerable
    - Applications that support "Auto-Update" / "Check for Updates" feature

Security Issues Involved With Developing and Deploying ActiveX Controls

- ActiveX Controls
  - Background
    - ActiveX controls are typically invoked using the browser. However, the browser is NOT a sandbox for ActiveX controls and as such they are like any other application executing locally on the machine
  - Discussion
    - Repurposing
    - Initialization from un-trusted source
    - Scripting powerful methods supported by the ActiveX control
  - Impact
    - Information Disclosure
    - Privilege Escalation
    - System Compromise
  - Exploitation / Mitigations
    - Remotely Exploitable

Security Issues Involved With Developing and Deploying ActiveX Controls

- ActiveX Controls
  - Recommendations
    - Digitally sign the ActiveX control
    - Use SiteLock Template
    - If not needed, do NOT mark the ActiveX control as safe for initialization and safe for scripting
- Identifying ActiveX controls security issues
  - Typical applications which are likely to be vulnerable
    - Applications using ActiveX controls

Security Issues Involved With Integrating and Leveraging Third Party Components

- Third Party Components
  - Background
    - Third party commercial off-the-shelf (COTS) components are frequently used by application development teams for purposes of rapid application development
    - For e.g. Openssl library for leveraging SSL; zlib library for compression
  - Discussion
    - Application teams tend to integrate and leverage the latest version of the COTS components available while integrating the COTS component into the application before the release
    - However, the COTS component vendors regularly release updates fixing the identified security vulnerabilities with the components.
  - Impact
    - Information Disclosure / Privilege Escalation / System Compromise / Denial of Service – Depends on the vulnerability
  - Exploitation / Mitigations
    - Local / Remote - Depends on the vulnerability

Security Issues Involved With Integrating and Leveraging Third Party Components

- Third Party Components
  - Recommendations
    - Subscribe to the third party COTS component security advisories and update the application to use the updated version of the third party component.

- Identifying third party component security issues
  - Typical applications which are likely to be vulnerable
    - Applications using third party components
    - Openssl and zlib library are two of the most widely third party components used by applications

Additional References

1. Socket Hijacking - Chapter 15, Writing Secure Code Vol. 2, Michael Howard et. al. ISBN: 0-7356-1722-8

2. Using SO_REUSEADDR and SO_EXCLUSIVEADDRUSE - http://msdn2.microsoft.com/en-us/library/ms740621(VS.85).aspx

3. Discovering and Exploiting Named Pipe Security Flaws for Fun and Profit - http://www.blakewatts.com/namedpipepaper.html

4. Named Pipe Security and Access Rights - http://msdn2.microsoft.com/en-us/library/aa365600(VS.85).aspx

5. CreateProcess() Security - http://msdn2.microsoft.com/en-us/library/ms682425(VS.85).aspx

6. ActiveX Security – Chapter 8, Hacking Exposed Web 2.0: Web 2.0 Security Secrets and Solutions, Rich Cannings, Himanshu Dwivedi, Zane Lackey et al. ISBN: 0-0714-9461-8

7. ActiveX Controls Security - http://msdn2.microsoft.com/en-us/library/bb250471(VS.85).aspx

Questions

Top 5 Security Issues Developers Don't Know About

Neelay S. Shah
Senior Security Consultant
Foundstone Inc. , A Division of McAfee
neelay.shah@foundstone.com

OWASP
March 13, 2008
NY/NJ Chapter

The OWASP Foundation
http://www.owasp.org