Event-driven system security vulnerabilities, an overview and demonstration (v1.1)

Simeon (simos) Xenitellis
Information Security Group,
Royal Holloway College,
University of London,
TW20 0EX United Kingdom
e-mail: simos74@gmx.net

HITB 2003, Kuala Lumpur
Agenda

- About event-driven systems
- Implementations
- Nature of vulnerability
- Timeline
- Demonstration
- Analysis & countermeasures
- Conclusion
What is “Event-driven”? 

- A method to model a software system. 
- Software system comprises of independent entities, executing at the same time. 
- Entities receive input from and through the system, not directly from producer. 
- Input is formulated as “events”. 
- “Events” are data structures, fixed size.
Example of EDS

- Modelling the Graphical User Interface (GUI).
- Applications execute at the same time, share input devices.
- System handles input devices, sends input as events to correct application.
- Event messages have “type” and “payload”.
Examples of EDS (cont.)

- Windows™, the X Window System and the Java Virtual Machine are such systems.
- “Notepad” is an EDS application.
- Mouse/Keyboard are input devices.
- Event messages contain an “event-type” and parameters.
EDS functionality analysis

An event-driven system object can be

- An application
  - Its windows or dialog boxes
    - their individual “controls” (buttons, text boxes...)

Objects may

1. **enumerate** other objects,
2. **send** events.
Practical example (1/2)

Available senders/recipients of events
Practical example (2/2)

Events In Notepad (writing “HITB”)
Nature of vulnerability

- You can interact with apps through events!
  - Events as attacking vector to overwrite memory
    - add to command line, getenv(), network socket, etc.
  - New avenue of attack, “crafted event sequence”
    - custom/crafted sequence of events

- Practical: Typically can send events to privileged apps!
Windows Security

Background

- Windows 2000, Windows XP and newer
- Can send events only to
  - same window station (since Windows 2000)
  - same desktop window
- But privileged apps may attach to default desktop
- Demonstration:
Window stations & desktops

- Default window station: WinSta0
- Default desktop window: Default
About type-safe languages

- About “type safety”
- “To protect your data types”.
- It’s about allowing to overwrite (or not) the memory (of the running application).
- Java, C#, etc are type-safe.
- C/C++ are not.
Overwrite and execute

- Overwrite memory
  - Buffer overflow
  - Format strings
  - ...events?
- Divert execution
  - overwrite instruction pointer/program counter
  - overwrite function pointer
  - ...events?
Timeline (’93)

- The **scrollbar bug**
- Crafted event sequence
- 10 years and counting!
- Verified: Win3.1, 3.11, 95, NT, 2K, XP...
- Demonstration follows.
Demonstration (scrollbar bug)

- Notepad.exe with sample text.
- Scrollbars are available.
Demonstration (scrollbar bug)

- Logitech MouseWare utilities.
- Enabled double-click for middle button.
- Still have that Logitech mouse :)

Event-driven system security vulnerabilities, HITB'03, KL
Demonstration (scrollbar bug)

- Click-Drag-DoubleClick on scrollbar.
- Note: DoubleClick is middle button.

1. Click
2. Drag
3. DblClick
Demonstration (scrollbar bug)

- Result: deformed scrollbar
Software reliability

- Programs exhibit defects
  - Inevitable (from software testing)
- When writing software,
  - we use software testing for catch defects
  - fix on reporting
  - some proactive measures
“fuzz” software testing (1/2)

- Kind of black-box testing
  - send garbage
  - observe
- University of Wisconsin, “fuzz” tool
  - against command-line apps
  - also event-driven systems
- See how often they fail (crash or hang)
“fuzz” software testing (2/2)

- Tried 3 times
- In ’95, on X applications
- In ’00, on Windows applications
- random input (keyboard/mouse): 1/5 fail
- random events: almost all fail
First exploit (Sep/2000)

- Events/attacking vector/buffer overflow
- By “DilDog”, now part of @stake
- Win2k affected
- Privileged app receives events
  - From current logged-on user
  - Buffer overflow
  - Exploit it
First study (May/2002)

- By yours truly
- Also announced at vuln-dev (securityfocus) on 6/July/2002
- Available at: http://www.isg.rhul.ac.uk/~simos/event-demo/
First study (May/2002)

- Events, a “neglected” input to software
- Can lead to security vulnerabilities
- Software security subset of software reliability
- Attack modes
  - Vector of attack, overwrite memory
  - “Pure event vulnerability”, specially crafted event sequence
Vulnerability requirements

- Attacker needs to be able to
  1. enumerate victim objects
  2. send events to victim

- Minimal requirements
What’s enumeration?

- Is obtaining a list (w/ descriptive information) of the event-driven system objects.
- Typically, one can enumerate
  - Applications
    - Their windows and dialog boxes
    - Their controls (like text boxes, buttons, etc.)
Sending events

- Is the sending of events to event-driven system objects. Choice of any event type/payload.
  - Windows: Can send events to any object available.
  - Java: only from same ‘CODEBASE’ and in same JVM.
  - X: From within same user account only, depends on value of xhost (default not allow)
### Matrix of requirements

<table>
<thead>
<tr>
<th>Condition\OS</th>
<th>Windows 9x</th>
<th>Windows NT/2K</th>
<th>Java</th>
<th>Java Plug-in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration</td>
<td>Yes</td>
<td>Yes</td>
<td>No**</td>
<td>No***</td>
</tr>
<tr>
<td>Sending</td>
<td>Yes</td>
<td>Yes</td>
<td>No**</td>
<td>No***</td>
</tr>
<tr>
<td>Interception</td>
<td>Yes</td>
<td>No*</td>
<td>No**</td>
<td>No***</td>
</tr>
</tbody>
</table>

*Only between objects belonging to the same user.

** Only for applets in same CODEBASE.

*** No, because each applet is executed in separate JVM.
Categorisation (1/3)

- Parameter semantics
  - Event parameters may contain references to risky data

- Example
  - WM_TIMER event parameter: Address to subroutine to execute
Categorisation (2/3)

- Sequence of events
  - Not all sequences are accounted for

- Example
  - Authentication scheme
  - Drag on path to authenticate
  - If state is stored on counter,
    - can lead to compromise
Categorisation (3/3)

- Interface-based security
  - The interface can be typically modified in order to let in custom data

- Example
  - Enabling/disabling window components
Infamous “shatter attack” (1/2)

- In August 2002 (one month later)
- Loads of publicity
- Events as vector of attack
- By Chris Paget
- Claim: Critical windows vulnerability
- Microsoft: not happy
- No referencing to any previous work
Infamous “shatter attack” (2/2)

- Overwrite memory:
  - Set the title bar with exploit code
- Divert execution:
  - Use WM_TIMER to execute code
More “shatter attacks”

- “shatter” stuck
- More vulnerabilities found
- On Windows
- Technical: Brett Moore @
  www.security-assessment.com
Technical on “SEH” attacks

- Overwrite memory
  - HDM_GETITEMRECT and similar
  - Writes objects geometry at address...
  - Set size, force to write geometry

- Divert execution
  - SEH: Structured Exception Handling
  - optional subroutine for post-execution
Demonstration

- Using events to target applications
- Not necessarily code execution
- Notion of privilege
Tool

- Identify objects and
- send events

Event-driven system security vulnerabilities, HITB'03, KL
Tool (under different credentials)
Retrieve password (yawtrap)

- RunAs dialog box
Retrieve password (yawtrap)

- HowTo
- But
  - unmask
  - select all
  - copy/paste
  - unselect
  - mask
Interface-based security
Event-driven system security vulnerabilities, HITB'03, KL
Interface-based security
Interface-based security
Interface-based security
Interface-based security
Interface-based security

Event-driven system security vulnerabilities, HITB’03, KL
Interface-based security
Interface-based security
Interface-based security

- Did we bypass license agreement?
- Depends (on country)
- Application that alters license screens?
- And more.. Business model.
- Like password managers
Disable applications

- Close personal firewall/antivirus
About WM_TIMER et al

- Payload: address to execute
- Demonstration:

  - Use MySpy to grab window handle of victim
  - Send WM_TIMER with address shown on victim.
  - Patched after “shatter” publicity.
Attacks against kiosks/etc (1/2)

Pay-per-view service

Event-driven system security vulnerabilities, HITB'03, KL
Attacks against kiosks/etc (2/2)

- Bypassing restrictions
Analysis (1/3)

- In event-driven system, no origin authentication for events
  - who sent this event?
  - impractical to include?
  - heritage of early Windows 3.x/9x
  - can do memory-based tricks (cheaper)
- Multi-user on same session
  - Memory protection, no event protection
Analysis (2/3)

- Two modes of exploitation
  1. Overwrite memory and divert execution
  2. Send crafted message sequence (state confusion)

- Type-systems can solve 1.
- Type-systems cannot solve 2.
- First exploit on 2?
Analysis (3/3)

- What’s privileged?
  - Application has something attacker wants
    - PGP
    - Outlook
    - Personal firewall / Antivirus

- TCPA and event-driven systems?
  - More apps become privileged
Lessons

- Typically no origin authentication
  - Cannot filter out bad apps
- System security
  - window stations/desktop windows, if possible
- Consider events as attack vector
  - Deploy kiosks/ATMs/security devices
  - Minimize inputs
- Privileged? It’s relative.
Lessons (2/2)

- Interface-based security?
  - Not available/don’t rely on.
- License agreement?
  - Reject unneeded events
- Multi-user on same session: not secure
  - i.e. Run application as “restricteduser”.

Event-driven system security vulnerabilities, HITB’03, KL
Summary

- Event-driven systems
- Implementations (mostly Windows)
- Nature of vulnerability
- Timeline
- Demonstration
- Analysis & countermeasures
Questions?