Adaptive Application Security Testing Model

Ashish Khandelwal
Gunankar Tyagi
Agendum

- What is Security Testing & its relevance?
- Traditional Security Testing Approach
- The Adaptive Application Security Testing Model
- Case Study - Application Product Testing Success Story
- Q&A
Security Testing

- A Quick Glance
- Security V/s Functional Testing
- Myths
Cost Impact

Monster.com suffered a heavy security breach in Aug 2007 that reportedly resulted in the theft of the confidential information for some 1.3 million job seekers.

The TJX Company breach, which was first reported in January of 2007, has been widely recognized as the largest reported theft of personal details ever lost by a company.

Operation Aurora affected as many as 2,411 companies and compromised data ranges from intellectual property, classified documents to credit card transaction details.

LOVE BUG exploited Microsoft Outlook e-mail client to execute programs. The damage resulting from this virus was reported to be in the billions of dollars.
A man who wants to remain rich, will make sure he locks his money in The “VAULT”
Case Study – Product Context Setting

- Application Product team
- Integrated Engineering team
- Skill Sets
- Specialized Skilled
Threat Model and its constraints

1. Identify Assets
2. Create an Architecture Overview
3. Decompose the Application
4. Identify and Prioritize them
5. Test and Record

Constraints:
- Complexity: Approach is time consuming as it involves listing and analyzing every possible asset in the product.
- Connectivity: Access to architectural diagram & specifically code is not always possible.
- Changeability: Agile methodology & environmental exposure can endanger threat model base.
Adaptive Ladder

Threat Model

Adversarial Security Testing

Peripheral Security Testing

Limited Access to Codebase

Expertise Level

Success Rate
Adaptive Model - Highlights

Two-tier sequential model

Peripheral Security Testing (PST)
Adversarial Security Testing (AST)

Each of these testing types is defined in terms of

- Inputs
- Activities
- Outputs

Adaptive Model kicks off with PST and then helps to

- Enhance security knowledge and experience
- Constantly Deliver results
- Build perquisite for AST
# Adaptive Model – Basic Workflow

## Adaptive Approach

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Activities</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA with Security Expertise</td>
<td>Identify Assets/Attack Base</td>
<td>Analysis, Research &amp; Result Documents</td>
</tr>
<tr>
<td>Code Access, Historical Knowledge, Security Experience</td>
<td>Execute &amp; Perform Sophisticated Analysis</td>
<td></td>
</tr>
<tr>
<td>AST</td>
<td>Identify EPs</td>
<td>Execution &amp; Results Document</td>
</tr>
<tr>
<td>QA with Attack Perspective</td>
<td>Document EPs</td>
<td></td>
</tr>
<tr>
<td>PST</td>
<td>Execute &amp; Perform</td>
<td></td>
</tr>
<tr>
<td>Expertise</td>
<td>Create Attack Model</td>
<td></td>
</tr>
</tbody>
</table>

**Abbreviations:**
- **ASTPST**: Adversarial Security Testing, Peripheral Security Testing
- **AST**: Adversarial Security Testing
- **PST**: Peripheral Security Testing
- **EPs**: Entry/Exit Points

---

Confidential McAfee Internal Use Only
Peripheral Security Testing

**Entry Points**
- Place where inputs are supplied to your application

**Exit Points**
- Desirable/undesirable output from the application.

**Outside Approach**
- Without much knowledge of internal implementation

**On the Surface**
- Easier to detect and require less effort.

**Tool Based**
- Peripheral Security Testing

**Outside-In approach**
- Entry Point Based

Confidential McAfee Internal Use Only
Sample Study I (Peripheral Security Testing)

- Files, Folders
- Registry Entries

Prepare Attack Model
- Information Disclosure
- Weak Permissions
- Buffer Overflow

Use Tools & Scripts
- FileMon
- Regmon
- ACL Editor

Execute & Record
- Write Results
- Document Issues

List Entry Points

Does your product functionality hamper if you deny the permissions to the temp folder?

Do the files (logs/event xml/binaries) contain sensitive data?

Is there a way in which you can cause buffer overflow in the file extension/file names?
Sample Study II (Peripheral Security Testing)

- **Identify Named Pipes (pipe list)**
- **View Permissions of Named Pipe (ObjSD)**
- **Check the product against Hijacking or Impersonating the Named pipe**

**Prepare Attack Model**
- Exploit weak permission
- Hijack the creation
- Impersonate the client

**Use Tools & Scripts**
- PipSec
- PipeList
- CreateAgentPipe
- ObjSD

**Execute & Record**
- Write Results
- Document Issues
## Peripheral Security Testing Checklist

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Entry/Exit Points</th>
<th>Attack Model</th>
<th>Tools/Scripts</th>
</tr>
</thead>
</table>
| 1     | File & Folders    | Information Disclosure  
(Weak Permissions  
Buffer Overflow)  | FileMon  
ACL Editor  
strings |
| 2     | Sockets           | Man-in-the-middle Attack  
Sniffing network traffic  
Send malicious data | Wire shark  
netstat.exe  
netcat |
| 3     | Registry Entries  | Registry Accessed by the product  
Permission of the registry keys | Regmon  
ACL Editor |
| 4     | Named Pipes       | Exploit weak permission  
Hijack the creation  
Impersonate the client | PipSec  
PipeList  
CreateAgentPipe  
ObjSD |
| 5     | User Interfaces   | Shatter Attack  
Format String Attacks | Shatter Tool  
WebText Convertor |
| 6     | Command Line Arguments | Exploit Undocumented command Line switches | Command Line switches /?, -?, /h, or -h.  
Process Explorer Image tab |
| 7     | Environment Variables | Uncover Environment Variables used by Product  
Manipulating data inside Product defined Environment Variables | Process Explorer Environment Tab  
System Environment Variable Tab |
| 8     | ActiveX Control   | ActiveX Repurposing Attacks  
ActiveX Fuzzing | COMRaider  
OLEView |
| 9     | Drivers           | I/O Verification  
Deadlock Detection  
Dangerous APIs  
Exceptions/Handlers/Memory  
Loading and Unloading Filter Driver  
Attach and Detach Filter Driver | Windows Utility-> Verifier.exe  
Windows Utility -> fltmc  
Microsoft Application Verifier  
Velocity Tool by Microsoft |
Attack Base

An entity of the product or the Operating System which can be Manipulated to perform an attack on software.

Observation Based

Gather past vulnerability information About the attack base.

Abuse Cases

Abuse cases (sometimes called misuse cases as well) are a tool that can help you begin to think about your software the same way that attackers do.
Sample Study III (Adversarial Security Testing)

- Complexity of ACL’s configuration
- Permissions cannot be assigned to all objects
- Exploiting Integrity Level (Vista specific)
<table>
<thead>
<tr>
<th>S.No.</th>
<th>Attack Base</th>
<th>Abuse Case Scenarios</th>
<th>References and Historical Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access Control List</td>
<td>Verification of apt ACL’s for your product resources</td>
<td>Shatter Attack</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Target NULL DACL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Look for dangerous ACE types</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-&gt; Everyone (WRITE_DAC)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-&gt; Everyone (WRITE_OWNER)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-&gt; Everyone (FILE_ADD_FILE)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Target Windows DAC weakness</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Target Windows MIC weakness</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Shell Extensions</td>
<td>List out shell extensions used by your product</td>
<td><a href="http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-5902">http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-5902</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>List the resources utilized by our Shell Extension</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Behavior of shell extension</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Effect of impersonating your product shell extensions</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Plugins</td>
<td>List resources used by BHO</td>
<td><a href="http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-2382">http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-2382</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Learn how to write a BHO</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Understand functionality of IE Plugin.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-&gt; This can give more attack vectors</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Effect of impersonating our product BHO</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Find a way to disable IE Plugin</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Denial Of Service</td>
<td>Do basic analysis of DOS Attacks</td>
<td><a href="http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1855">http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1855</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identify the services rendered by your product</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identify the ports used by the services</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identify tools to send specially crafter packets to perform a</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOS Attack on our product. ( use historical info )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Analyze the results</td>
<td></td>
</tr>
</tbody>
</table>
AASTM Model - Recap

- Position yourself on the Adaptive Ladder and then design your security testing strategy.
- The idea is to find security defects in the product. A model is important but not a constraint.
- Follow Peripheral and Adversarial approaches as a guideline to target security flaws.
- Creating a dedicated Skill-set base within the team helps a lot.
- Even if it’s an ad-hoc approach it’s good to expose some security shortcomings.
References

- http://en.wikipedia.org/wiki/Operation_Aurora
- http://en.wikipedia.org/wiki/Mandatory_Integrity_Control
- http://nikkigsblog.files.wordpress.com/2010/04/locked-house.jpg
For any help/query

Ashish_Khandelwal@Mcafee.com

Gunankar_Tyagi@McAfee.com