Securing Legacy Apps – Worth the Effort?
Michael Solomon
29 June 2016
Yes. Well, it depends.

It depends on thoughtful assessment of your application and its vulnerabilities.
Agenda

- What is a legacy application?
- How are they vulnerable?
- What can we do about it?
- Where can I get help?
You do have AT LEAST one legacy application, right?
Legacy security often relied on physical barriers.
Physical controls protected data.
These are just the most common reasons that legacy applications have security vulnerabilities.
Legacy client types

**Terminal systems**
- Mainframe
- TTY terminals
- OpenEdge Character (CHUI) client

**Client/Server**
- Workstation
- Networked connection to database
- OpenEdge GUI client

**Browser**
- Internet application
- Runs in web browser
- OpenEdge WebSpeed
Terminal system application vulnerabilities

- Authentication often depends on OS
- Authorization based on file system permissions
- *** Limited or nonexistent encryption capability ***
  in-transit/at rest
- No local input data validation
- Often dependent on physical terminal characteristics (perhaps for input validation)
- Due to historical physical separation security
  • other areas may be sloppy (i.e. hardcoded userids/passwords)
Client/Server application vulnerabilities

- Some applications offload too much data to clients for efficiency
- Server DB is exposed to non-authorized connections
- Deployment can be problematic and slow down distribution of patches
- Potential vulnerabilities for older Client/Server software
- Session hijacking
- Harder to administer well
  - Many insecure defaults are in the wild
Browser application vulnerabilities

- Session hijacking
- Exposure of server components (DB/AppServer)
- Higher reliance on network/server security
- More difficult to handle per-call authentication/authorization
What does it take to secure legacy applications?

- Inventory all legacy applications
  - Description
  - Version in use
  - Dev history
  - Footprint
  - Requirements
  - Responsible person

- Risk assessment (limit scope for expediency if necessary)
  - Sensitive data access
  - Business function criticality
  - Compliance
  - Attack surface

- In-depth analysis of "at high risk" systems

- Mitigate high priority risks

Progress
Mitigation options

1. Do nothing
2. Harden surrounding environment → Threat modeling
3. Harden application → Threat modeling
4. Replace application
- May be the best choice
- Risk assessment provides response recommendations
- Better than wasting resources on ineffective controls
- Choose this option
  - Don’t default to it
Manage risk with threat modeling
How to threat model
What are you building?

What can go wrong?

What are you going to do about it?

Check your work
What are you building?

- **Create a system model**
  - Abstracts away the details

- **Diagrams are key**
  - Mathematical models are rare in industry

- **Primary focus in threat modeling**
  - Data flows
  - Threat boundaries

- **Common diagram types**
  - Data Flow Diagrams (DFD)
  - Swim Lanes
  - State machines
Developed in the early 70s, and still useful

Simple: easy to learn, sketch
Threats often follow data

Abstracts programs into:

Processes: your code
Data stores: files, databases, shared memory
Data flows: connect processes to other elements
External entities: everything but your code & data. Includes people & cloud software
Trust boundaries now made explicit
Fun to brainstorm
Mnemonics, trees or libraries of threats can all help structure thinking
Structure helps get you towards completeness and predictability
STRIDE is a mnemonic
  Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
Easy, right?
<table>
<thead>
<tr>
<th>Threat</th>
<th>Property Violated</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spoofing</td>
<td>Authentication</td>
<td>Impersonating something or someone else.</td>
<td>Pretending to be any of Bill Gates, Paypal.com or ntdll.dll</td>
</tr>
<tr>
<td>Tampering</td>
<td>Integrity</td>
<td>Modifying data or code</td>
<td>Modifying a DLL on disk or DVD, or a packet as it traverses the network</td>
</tr>
<tr>
<td>Repudiation</td>
<td>Non-repudiation</td>
<td>Claiming to have not performed an action.</td>
<td>“I didn’t send that email,” “I didn’t modify that file,” “I certainly didn’t visit that web site, dear!”</td>
</tr>
<tr>
<td>Information Disclosure</td>
<td>Confidentiality</td>
<td>Exposing information to someone not authorized to see it</td>
<td>Allowing someone to read the Windows source code; publishing a list of customers to a web site.</td>
</tr>
<tr>
<td>Denial of Service</td>
<td>Availability</td>
<td>Deny or degrade service to users</td>
<td>Crashing Windows or a web site, sending a packet and absorbing seconds of CPU time, or routing packets into a black hole.</td>
</tr>
<tr>
<td>Elevation of Privilege</td>
<td>Authorization</td>
<td>Gain capabilities without proper authorization</td>
<td>Allowing a remote internet user to run commands is the classic example, but going from a limited user to admin is also EoP.</td>
</tr>
</tbody>
</table>
Using STRIDE

How can each STRIDE threat can impact each part of your model

• How could an attacker tamper with this part of the system?

Make it easier

• Elevation of Privilege Game
• Attack Trees
• Experience

Track issues as you find them

• Track assumptions too
What are you going to do about it?
Threats and assumptions

For each threat
- Fix – remove functionality
- Mitigate
- Accept – Be careful about accepting customer risk
- Transfer – License agreements, TOS

For each assumption
- Check
- Reconsider wrong assumptions
## Ways to mitigate threats

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation Technology</th>
<th>Developer Example</th>
<th>Sysadmin Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spoofing</td>
<td>Authentication</td>
<td>Digital signatures, Active directory, LDAP</td>
<td>Passwords, crypto tunnels</td>
</tr>
<tr>
<td>Tampering</td>
<td>Integrity, permissions</td>
<td>Digital signatures</td>
<td>ACLs/permissions, crypto tunnels</td>
</tr>
<tr>
<td>Repudiation</td>
<td>Fraud prevention, logging, signatures</td>
<td>Customer history risk management</td>
<td>Logging</td>
</tr>
<tr>
<td>Information disclosure</td>
<td>Permissions, encryption</td>
<td>Permissions (local), PGP, SSL</td>
<td>Crypto tunnels</td>
</tr>
<tr>
<td>Denial of service</td>
<td>Availability</td>
<td>Elastic cloud design</td>
<td>Load balancers, more capacity</td>
</tr>
<tr>
<td>Elevation of privilege</td>
<td>Authorization, isolation</td>
<td>Roles, privileges, input validation for purpose, (fuzzing*)</td>
<td>Sandboxes, firewalls</td>
</tr>
</tbody>
</table>
Check your work

Quality assurance
Check that you covered all the threats & assumptions
Check that each is covered well
Sometimes you just can't mitigate enough vulnerabilities in legacy applications.
Replace software when mitigation cost is too high and replacement software meets needs
The new Solomon Consulting portal coming soon.

Ready  Set  Secure

www.solomonconsulting.com
References

- http://www.codingthearchitecture.com/2015/03/07/security_concerns_for_legacy_systems.html
- https://www.owasp.org/index.php/OWASP_Supporting_Legacy_Web_Applications_in_the_Current_Environment_Project