

Design & Operational Review

Penetration Testing & Vulnerability Analysis

Brandon Edwards

brandon@isis.poly.edu

Agenda

- What are Design Flaws?
 - Example Design Flaws
 - Performing Design Review
 - Setting up Implementation Review
-
- What are Operation Security concerns?
 - Example operational security concerns
 - Thinking about operational security

Design Flaws

What are design flaws (vulnerabilities)?

- “A design vulnerability is a problem that arises from a fundamental mistake or oversight in the software’s design” – The Art of Software Security Assessment
- Lack of security forethought
- Intrinsic to the app’s architecture
- Can be subtle
- Are often devastating

Design Flaws = Ownage

- Reliable!
 - Work seamlessly across OSes/versions
 - Do not corrupt memory!
 - Usually have little/no preventative protection mechanisms
 - Rarely alarm detection tools
 - Do not crash on failure 😊
- Easily fingerprinted
- Work well in symphony
- Difficult to fix, so they hang around for a while 😊

Design Flaws Commonly Affect

- Authentication
 - Bypasses
 - Lack of Authentication
- Authorization
 - Lack of Authorization
 - Giving higher privileges to lower privileged users
- Cryptography
 - Storing key with plaintext
 - Improper use
- Business Logic

Design Flaws Examples

- RealVNC NULL Authentication Method, Authentication Bypass in 2006

```
switch(authentication_type)
{
    case USE_PASSWORD:
        getCredentials(&creds.password);
        if (validateCreds(creds))
        {
            auth_flag = 1;
        }
        break;

    case USE_NULL_AUTH:
        auth_flag = 1;
        break;
}
```

Design Flaws Examples

- Vendor inserted Backdoors
 - Barnaby Jack's ATM Jackpotting attack
- Debugging/QA hardcoded usernames/passwords

```
if (!strcmp(password, "8==D"))
{
    auth_flag = 1;
    break;
}
```

Design Flaws Examples

- Authorization oversights
 - Java deserialization vulnerabilities

```
try {
    ZoneInfo zi = (ZoneInfo)

        AccessController.doPrivileged(
            new PrivilegedExceptionAction() {
                public Object run() throws Exception {
                    return input.readObject();
                }
            });

    if (zi != null) {
        zone = zi;
    }
} catch (Exception e) { }
```

Design Flaws Examples

- Authorization oversights
 - Checks not performed on portions of a web application (forced browsing)
- Cryptographic misunderstandings
 - Hardcoded keys
 - Storing/Sending key with cipher text

Finding Design Flaws, Steps

- Collect Information
- Model the Application
- Understand Business Logic, Intention & Architecture
- Use Above to Target Areas
- Busticate the Application

Information Collection

- Application Specs, RFCs, etc
- Architectural Documentation
- Data Flow Diagrams
- *Especially anything on security relevant matters*

Modeling the application

- **Identify Resources**

Resources are anything used by the application which may be useful to an attacker

- Data (Creds? Personal info?) used by the application
 - Control the application has or grants access to
- All of resources combined represent every piece of access to data or functionality offered by a system

Modeling the application

- **Examine Input**

- What type of input does the application get?
- Where/Who does the data come from?
- What is the purpose of the input?
- Track the input data flow
- How trusted is the data?
- How much validation is performed on the data?

Modeling the application

- **Something to think about**

Any external influence you can provide which affects the program is input!

Modeling the application

- **Examine Output**
 - What type of output data is there?
(Is it sensitive?)
 - Where does the output go?
 - How does the input result in output?
 - How can the data be leveraged by an attacker?

Modeling the application

- **Something to think about**

Any observation of a program response that can be measured is output

Modeling the application

- **What type of External Components are there?**
 - Network Services used by the application?
 - Libraries?
 - Databases?
 - Anything the application relies on that is not part of the application itself

Modeling the application

- **User Roles**

- What are the various roles of users?
- What types of users can exist?
- How do they identify themselves?
- How is a user authenticated?
- How are privilege levels defined?
- Where are user roles unclear?

Modeling the application

- **Trust Boundaries**

- Given the user roles, and the resources, where should trust boundaries lie?
- Where is trust granted by the application to external components
- Are these trust boundaries enforced uniformly?
- Are there unclear areas of trust?

Modeling the application

- **After reviewing each area, combine your observations**
- **Use the above to identify targets**
- **Yes, it's cliché, but "Think Like an Attacker"**
 - Where was security not considered?
 - Where do the security considerations fall short?
 - Edge cases?
 - Logical flaws?
 - Think about what the app can do beyond what it was intended to
 - How can the application deviate from the intent?

Modeling the application

- Review the target components
- Think about the logic of the application
- Try and think beyond the intentions of the developer
- Even if no architectural or design flaws are found, take notes and keep the information...

All of the gathered information will be useful for implementation reviews!!!

Operational Review

What are operational flaws (vulnerabilities)?

- “Operational vulnerabilities are the result of issues in an application's configuration or deployment environment. ” – The Art of Software Security Assessment
- Infrastructure supporting the application
- Environmental concerns

Operational Concerns

- Operational issues happen outside the scope of the application
- Not problems which will be indicated in the source itself
- Possible concerns can be inferred by thinking about what the application does, and how it is likely deployed

Operational Review

- Poor configuration?
- External components...
 - Databases setup with open ports, default passwords
 - Default scripts left around web servers
 - Anonymous logins allowed on ftp servers
 - Unpatched components
- Failure to use secure technologies around the application's deployment
 - Lack of SSL on web servers exchanging creds
 - Poorly generated certificates/bad random data (Debian is an example)

Operational Review

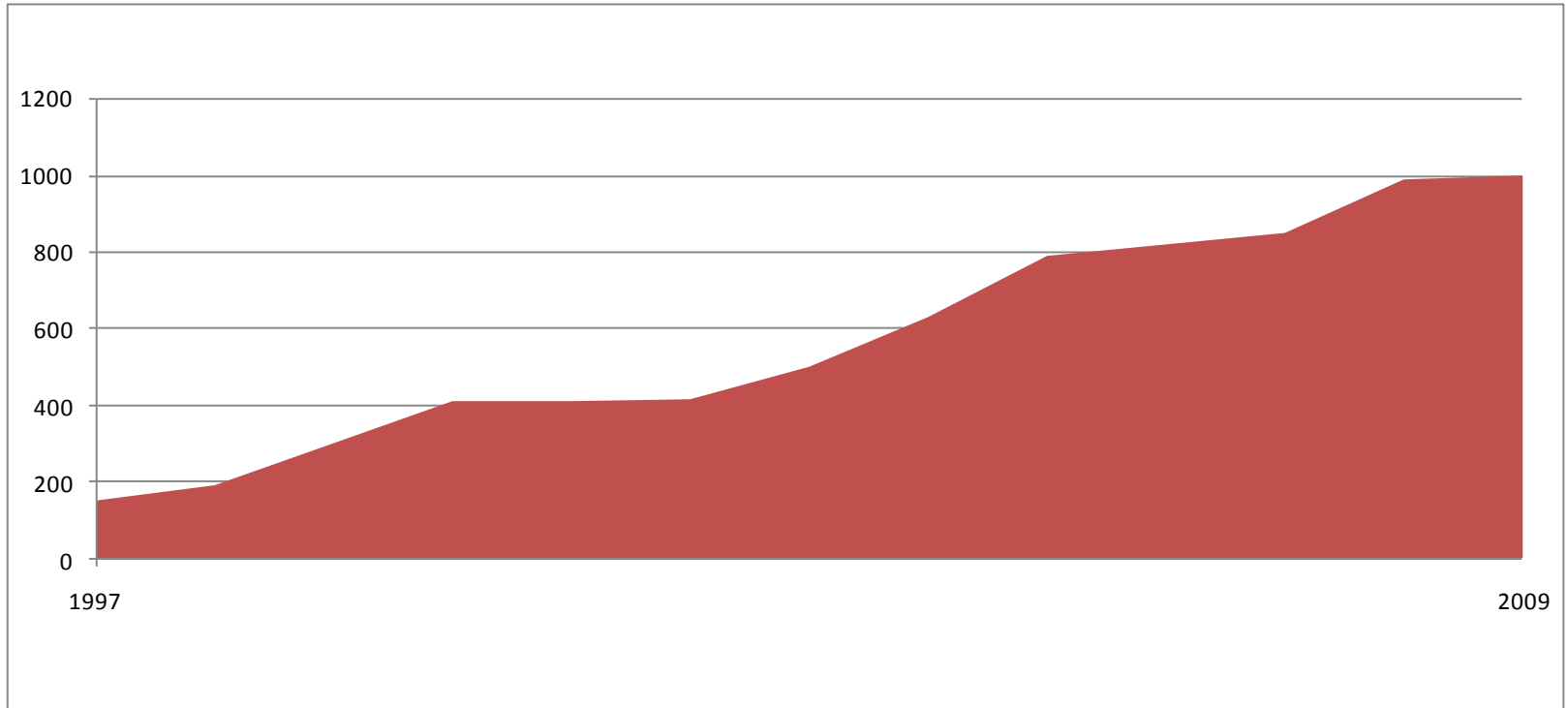
- Security Hardening Mechanisms
 - DEP
 - Stack Cookies
 - Hardened Heap
 - SEH validation
 - Address Space Layout Randomization
- Policy, security in depth
 - Network segmentation
 - System user roles
 - Chroot jails
 - Firewalls, application firewalls

Operational Review

- Think about the application is probably deployed
- How can it be abused?
- What do admins notoriously fail at in this way?

Graph Slide that shows NOTHING

LOL THIS IS MEANINGLESS



This data is useless

Questions?