Securing the Data Center
Part I – Data Center Security Model

Yohay, Shachaf, Spring 2015
Information provided in these slides is for educational purposes only.
Outline

1. Introduction
   • Why Security matters?
   • Security control model and definitions
2. Security challenges (deep-dive)
   • Identities management
   • Applications security
Why Security matter

• Security is everywhere, but in thought only when something happens

• A Security breach can cause real money lose, reputation, intellectual property lose and legal actions
Some Security breaches from 2014

PlayStation Network, Xbox Live hit by DDOS attacks

By Brendan Sinclair


Target Puts Data Breach Costs at $148 Million, and Forecasts Profit Drop

By RACHEL ABRAMS AUG. 5, 2014


Sony Delays Earnings Report Following Sony Pictures Hack

By Aaron Mammit, Tech Times | January 25, 1:13 AM

Data Center Security

• Data Center is a complex environment
  • Many interacting components

• Securing the datacenter is complex
  • Each component exposes its own vulnerabilities
  • Interaction between components introduces additional vulnerabilities

• Security controls will be discussed
  • Divide the data center into layers
  • Each layer is composed of attack vectors, vulnerabilities and controls
Definitions

• **Attack surface:** The sum of the different points (the "attack vectors") where an unauthorized user (the "attacker") can try to enter data to or extract data from an environment

• **Vulnerability:** A weakness which allows an attacker to reduce a system's information assurance. Vulnerability is combined of three elements: a system flaw, attacker access to it, attacker capability to exploit it

• **Threat:** A possible danger that might exploit a vulnerability to breach security

• **Risk:** The potential for loss, damage or destruction of an asset as a result of a threat exploiting a vulnerability.

• **Security Controls:** Safeguards or countermeasures to avoid, counteract or minimize security risks
Security Definitions: example of an attack synopsis

https://www.owasp.org/index.php/Top_10_2010-Main
Security Control Model: Controls summary

Applications
Information/Data
Management & Identity
Network
Compute & Storage
Physical
Data Center Security Model (2/4)

• Applications
  • Layered architecture very common in the data-center
    • Client → Web site → Application logic → Database
  • Each layer exposes vulnerabilities
  • Attacks range from stealing / manipulating data to Denial of Service (DoS)
  • We will discuss this in depth later

• Information / Data
  • Storage options that can be hard drives, files, objects and databases
  • the crown jewels of the business - mainly what attackers are after
  • Should be protected using access control, encryption, data lifecycle management
Data Center Security Model (3/4)

• Identity and authentication
  • Identities form the basis for security in the data center
  • Identity management is a **big** issue
    • Authentication, authorization, etc.
  • Many attacks begin by forging identities
  • **We will discuss this in depth later**

• Network
  • The basic way systems interact, hence the gate to computer resources
  • Requires network security controls to control access
    • Firewalls, Access Lists, Networks Separations such as Demelitrized Zone (DMZ) etc.
Data Center Security Model (4/4)

• Compute and storage
  • Encompass hardware attacks, OS, hypervisor, etc.
    • e.g., manipulate the hypervisor to perform a VM Escape
  • **Big** concern for the data center (e.g., Amazon)
    • Usually requires special expertise

• Physical
  • Measures designed to deny unauthorized access to facilities, equipment and resources
    • CCTV surveillance, security guards, barriers, etc.
  • Assumption – if someone has physical access to a resource \(\rightarrow\) Game over

Photo: Intel® facebook
Part I – Security challenges (deep-dive)

Identity and Access Management
Identity Access Management (IAM) Overview

• Defines the methods, processes, and capabilities used to enable secured access to resources
  1. **Identification** - method used for ensuring that a subject is the entity it claims to be (user, process)
  2. **Authentication** – the process of actually confirming that identity using credentials (password)
  3. **Authorization** – the privileges granted to a subject when performing an action over a resource (read/write)
Identity

- The digital data that uniquely describes an Entity such as person or a thing, and contains attributes of it (ID, Name, Address)

The problem

Traditionally enterprise identities were created on premise, e.g., Active Directory

- We now have many identities:
  john.doe@enterprise.com; john.doe@gmail.com; jd@outlook.com
- protecting those identities is in the hand of each provider
- Each identity has its own password → multiple passwords and repeated passwords across services
- Identity theft becomes an issue
- In the data center, Privileges accounts (administrator) also poses a risk
A solution to multiple identities - Federation (e.g. SAML)

• Allows secure web domains to exchange user authentication and authorization data
• Using SAML, an online service provider can contact a separate online identity provider to authenticate users who are trying to access secure content.
• SAML only supports web browsers, hence not applicable for some native apps
• A Single-Sign-On solution as the identity provider (username+password) remains the same across environments.
• Pre-setup required: exchanging keys
SAML flow

1. User accesses URL in app
2. HTTP POST to AS w/ Auth Request
3. User is sent to login page at AS
4. User logs in
5. Redirect to app w/ SAML token
6. User is logged in to resource server

http://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/
Authentication

The process for confirming the identity of a subject

Multi factors of authentication:

- knowledge factors ("what you know"), such as passwords
- possession factors ("what you have"), such as ATM cards, certificate
- inherence factors ("what you are"), such as fingerprint

A combination of 2 from the above will be considered “good” enough

Bundling the 2 or more factors into a single process will harden the authentication
knowledge factors: Passwords

• Brief survey
  1. how many password you have
  2. how many using the same password
  3. how many has those in a file/document
Password attacks

**Password Guessing:** brute force, dictionary, hybrid password

**Password Resetting:** bypassing password guessing mechanism: request a password reset on the user behalf, bypass the operating system boot

**Password Cracking:** hash guessing, rainbow tables

**Password sniffing & capturing:** sniff the password on the fly or by malware on a client

**Best method:** just ask (social engineering)

**Risk:** Identity theft, data loss, Systems tampering (privileged accounts)
**Authentication attacked Example:**

**Password Hash Suite**

- Windows saves its password in a hash
- Hash Suite will also pull local windows password and reveal it in a few hours or more using a dictionary, rainbow tables etc.

<table>
<thead>
<tr>
<th>Username</th>
<th>Hash</th>
<th>Cleartext</th>
</tr>
</thead>
<tbody>
<tr>
<td>hackis</td>
<td>089395F62E60087A0AD3B4355B144EE</td>
<td>NURSE</td>
</tr>
<tr>
<td>emeesees</td>
<td>EE0D67683CE23AEF509844355B168930</td>
<td>???????????L</td>
</tr>
<tr>
<td>mpascerl</td>
<td>0F7AE80D2705511A818318E428188</td>
<td>????????????</td>
</tr>
<tr>
<td>nwieder</td>
<td>3F7F2451335900F93E2847538002486</td>
<td>????????????</td>
</tr>
<tr>
<td>lrotthmen</td>
<td>441E593D999B81B1CAAD3B4355B144EE</td>
<td>????????????</td>
</tr>
<tr>
<td>nplace</td>
<td>54D53524B8CE978EDF12B82D0D2B8ADB7</td>
<td>????????????</td>
</tr>
<tr>
<td>gl00</td>
<td>6F63380A833111EAAAD3B4355B144EE</td>
<td>????????????</td>
</tr>
<tr>
<td>glibby</td>
<td>7B7F2358972AE3117AD3B4355B144EE</td>
<td>????????????</td>
</tr>
<tr>
<td>gnjani</td>
<td>8845CF5578CE69950AD3B4355B144EE</td>
<td>2000</td>
</tr>
<tr>
<td>itarmom</td>
<td>5B16CF9298933A9AD3B4355B144EE</td>
<td>3D65ES</td>
</tr>
<tr>
<td>ilenni</td>
<td>7F8461B89C8D92B0A3D3B4355B144EE</td>
<td>????????????</td>
</tr>
<tr>
<td>ilellinghay</td>
<td>899E5D0D34DE9G6AAD3B4355B144EE</td>
<td>????????????</td>
</tr>
<tr>
<td>tschullay</td>
<td>E3A6281709E2613AA3D3B4355B144EE</td>
<td>GIGI</td>
</tr>
<tr>
<td>trando</td>
<td>34F1DC8252E9AD09A3D3B4355B144EE</td>
<td>????????????</td>
</tr>
<tr>
<td>osfreve</td>
<td>84D6C965A56292E29F8284727B733AB1</td>
<td>????????????</td>
</tr>
</tbody>
</table>

*Image of Hash Suite interface showing password hashes and cleartexts.*
Controls

• Password policy: minimum length, change internal, combination, no words, number of failed login attempts
• Enforce privileged accounts for data center management
• Store and pass passwords in an encrypted way (https)
• Apply Single-Sign-On (SSO): The ability to log in once in once and gain access to all systems (e.g. SAML).
• Don’t pass the password: Kerberos (pass tokens)
• User awareness
• Use multi factor authentication
Possession factors ("what you have")

category of user authentication credentials based on items that the user has

• Physical Possession factors
  • a hardware device such as a security token or a USB
  • Magnetic cards

• Logical Possession factors
  • Text message to an email address
  • Certificate
  • Mobile applications
Possession factors attacks

1. An attacker can **steal** the possession factor.
2. An attacker can **copy** the possession factor.
3. The attacker can **intercept** the authentication process.
4. **hijack** access after authentication.
5. An attacker can **determine** the shared secret using reverse engineering for example.
Certificates - A logical Possession factors

• The certificate includes information about the key, information about its owner's identity, and the digital signature of an entity that has verified the certificate's contents are correct.

• If the signature is valid, and the person examining the certificate trusts the signer, then they know they can use that key to communicate with its owner.

• In a public-key infrastructure (PKI) scheme, the signer is a certificate authority (CA) that is trusted.

• Certificates are used for authentication and also for signing of applications, code and content.

• Example of an attack:
  Flame malware used a fake certificate to sign its code and be a valid application.
Certificates - Possession factors cont.

• The are 2 types of client side authentication certificates
  • User certificate – contains information about the user, e.g., user name and sometimes authorization information, e.g., the user is allowed to…
  • Device/machine certificate - contains information about the specific device, e.g., device name, and may contain information for validation ,e.g., MAC address

• Pros
  • Considered as strong authentication because of it is not easy to export it
  • Industry standard and supported protocols to support protection and revocation

• Cons
  • Static - can be stolen and used elsewhere
  • Hard to manage and to provision across OSs
Possession factor: OTP (One Time Password)

**The problem:** Certificates and passwords can be stolen and they are static.

**A solution:** Provide a possession factor that can’t be replayed

**One-Time-Password**
- A one time password algorithm, that once used, becomes invalid.
- The password (usually a number), is being passed to a secondary system
  - Text message to a phone
  - Client application
  - Phone call or email
- used as part of a 2 factors authentication method
- Pros: Easy to deploy and used across OS’s; Not vulnerable to replay attack
- Cons: No bundling to a specific device; Requires key management (revoking physical keys)
Log On

Windows
User name: Gerlnmaman1
(domain/idsid)
Password: ************
Submit

Please enter your onetime password

Submit

Text from +41 79 807 30 40
One Time Password: 687315
Close  Reply

access the application
Controls

- Change the keys more frequently
- Limit the time a key is valid for
- Manage token and revoke when needed
- Encrypt the traffic passing the tokens in
- Logs and business intelligence detecting anomalies (Two token used in different location)
- User awareness
Inherence factors ("what you are"): Biometrics

- **Problem**: something you know or something you have factors can be stolen or spoofed, and they are only a reference to an identity
- Biometric authentication is the physical identity of a person
- Biometric types: finger, eye scan, eye retina, face, Voice, DNA
  - Behavior analysis: keystrokes prediction, signature.

- **Pros**
  - Difficult to spoof - unique identifier specific to the individual
  - Some of the biometrics never change (fingerprint/Iris)

- **Cons**
  - False acceptance rate/False rejection rate - the probability that the system correctly/incorrectly matches the input pattern.
  - Identity theft of a person’s biometric or the database storing it - a Fingerprint Cannot be reset if stolen
  - Biometrics are changing - A sick person has a different tone of voice.
## Authentication factors - summary

<table>
<thead>
<tr>
<th>Factor</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Something you know (passwords)</td>
<td>- The easiest method today for authentication</td>
<td>- Relatively easier to hack using social engineering, theft, sniffing, guessing</td>
</tr>
<tr>
<td></td>
<td>- Easily understood by users</td>
<td>- Users’ need to remember too many passwords</td>
</tr>
<tr>
<td></td>
<td>- Low cost</td>
<td>- Cost of managing and supporting it</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Used for privileged accounts (high value)</td>
</tr>
<tr>
<td>Something you have (Certificates, Tokens, OTP)</td>
<td>- Provides a physical proof when used in a combination with a password, considered strong</td>
<td>- Can be stolen and or hacked</td>
</tr>
<tr>
<td></td>
<td>- Can be revoked</td>
<td>- Cost of managing and operating (SMS/Token)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Some are more susceptible to reply attacks</td>
</tr>
<tr>
<td>Something you are (Biometric)</td>
<td>- Difficult to spoof</td>
<td>- False acceptance rate/False rejection rate</td>
</tr>
<tr>
<td></td>
<td>- Non intrusive (most of them)</td>
<td>- Identity theft of a person’s biometric or the database storing it</td>
</tr>
<tr>
<td></td>
<td>- Some of the biometrics never change</td>
<td>- Biometrics are changing</td>
</tr>
</tbody>
</table>
Identity Access Management (IAM) - Summary

**Identification**: a method of ensuring that a subject is the entity it claims to be (user, process)

**Authentication**: process of actually confirming that identity using credentials (password)

**Authorization**: the privileges granted to a subject (read/write) when performing an action over a resource

**Single-Sign-On (SSO)**: authentication done once then access multiple resources
Attack - Real life example

Think like a hacker - if you would want to get it - what would you do?
How would you prevent it?
Part II – Security challenges (deep-dive)

Applications
Applications Overview

• Applications types:
  • Native application (Desktop vs Mobile).
  • Web application.
  • Web sites.
  • Web services.
  • Client-Server distributed applications. *(Has been spoken during the course)*

• The application layer is the way systems and people interact with the data center resources.
• Hence, an area of concern for developers, designers and programmers.
3 Tier Application Architecture
Application Security Overview

• **Measures** taken throughout the code's life-cycle to prevent attacks

1. **Websites attacks**
   - Code injections - Attacker injects code into computer program to change the course of execution.
   - Web manipulation - Attacker manipulate your Website for their own purposes.

2. **Web services attacks**
Websites Attacks - Code Injections

• **SQL injection:**
  • Entering input of SQL language into vulnerable input areas in a database driven application.
  • any database application is a target.

• **LDAP injection:**
  • What is it? The **Lightweight Directory Access Protocol (LDAP)** is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet protocol (IP) network.
  • It’s possible to modify LDAP statements using a local proxy.
Websites Attacks - Code Injections cont.

• **Xpath injection:**
  • What is it? Querying XML is done with XPath, a type of simple descriptive statement that allows the XML query to locate a piece of information.
  • Xpath injection is being done by sending intentionally malformed information into the web site, an attacker can find out how the XML data is structured, or access data that he may not normally have access to.

• **OS commands**
  • The goal is execution of arbitrary commands on the host operating system via a vulnerable application.
SQL Injection

**Developer**: select * from users where mail='john.doe@something.com' and pass='1234'

**Hacker**: select * from users where mail='john.doe@something.com' or 1=1 -- and pass='1234'

**Outcome**: Hacker log in without actually having a password

**Developer**: select * from users where mail='john.doe@something.com' and pass='1234'

**Hacker**: select * from users where mail='john.doe@something.com'; shutdown --

**Outcome**: Hackers shuts down the database
SQL injection Cont.

How to avoid such attacks:

• **Prepared statements:** Force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied.

• **Stored Procedures:** SQL code for a stored procedure is defined and stored in the database itself, and then called from the application.

Both of these techniques have the same effectiveness in preventing SQL injection so your organization should choose which approach makes the most sense for you.
LDAP Injection

**OWASP Example:**
```java
<input type="text" size=20 name="userName">Insert the username</input>
String ldapSearchQuery = "</cn=" + $userName + ">");
System.out.println(ldapSearchQuery);
```

• **The Attack:**

If the variable $userName is not validated, might accomplish LDAP injection

• If a user puts “*” on box search, the system may return all the usernames on the LDAP base

• If a user puts “jonys) ( | (password = * ) )”, it will generate the code below revealing jonys’ password ( cn = jonys ) ( | (password = * ) )
OS commands

- Injecting commands to the underlying operating system from the URL:
  
  `www.mysite.com/viewcontent.php?filename=my_great_content.txt;ls`

  - the attacker injected the ‘ls’ command which will give him the ability to list the files of the directory the code is running on.

- Common injection pattern and result:
  - `shell_command` - executes the command
  - `${shell_command}` - executes the command
  - `| shell_command` - executes the command and returns the output of the command
Application Security Overview

• **Measures** taken throughout the code's life-cycle to prevent attacks
  
  1. **Websites attacks**
     - Code injections - Attacker injects code into computer program to change the course of execution.
     - Web manipulation - Attacker manipulate your Website for their own purposes.
  
  2. **Web services attacks**
Websites Attacks - web site manipulation

• **Clickjacking**
  - Attacker overlays multiple transparent or opaque layers to trick a user into clicking on a button or link on another page.
  - Using only CSS and HTML, an attacker can create a transparent IFRAME of a victim web page that contains privileged button
Websites Attacks - web site manipulation (self Learn)

- **XSS(Cross Site Scripting)**
  - Web browsers execute code sent from websites (HTML, JS, flash). The attacker is using the website to forward an attack.
  - **How?** the input is embedded into the HTML output.
  - **Input?** input is coming from: Form, Cookies, DB and server vars.

**Summary**

The browser can be easily tricked to perform operations without the user’s knowing anything about it.
ClickJacking - trick a user into clicking on a button
Application Security Overview

• **Measures** taken throughout the code's life-cycle to prevent attacks

1. **Websites attacks**
   - Code injections - Attacker injects code into computer program to change the course of execution.
   - Web manipulation - Attacker manipulate your Website for their own purposes.

2. **Web services attacks**
2. Web Services Definition

**Wikipedia**
A method of communication between two electronic devices over a network. It is a software function provided at a network address over the Web with the service *always on* as in the concept of utility computing.

**WC3**
A software system designed to support *interoperable* machine-to-machine interaction over a network.
2. Web Services attacks

Programmer's Assumptions:
• Most (good :) ) programmers tend to think that web services are unattackable.
• Tend to invent security controls in order to create security for their web services.

Security overview:
• Common programming model for security independent of transport.
• Security is incorporated at the message layer rather than the transport layer
• Firewall friendly
• Persisted messages may support non-repudiation.
• Exposing system functionality over the internet
• Consuming services from third party organizations.
• Propagating user’s identities across application boundaries.
2. Web Services attacks

**Web Services design:** *We poke holes in your firewall so you don’t have to!*
2. Web Services attacks

Application side

```javascript
$http.get("http://localhost:1337/applicationsName")
  .success(function(data){
    $scope.apps = data;
    $scope.loadingAnimationActive = false;
    $log.info($scope.apps);
  });
```

Web Service side

```javascript
getApp: function(req, res) {
  // req.session.authenticated = false;
  // res.clearCookie('uiicc');
  // return res.json(200,{success: true});
  console.log('getApp method has been called');
  console.dir(req.query.AppName);
  var connection = new sql.Connection(sails.config.mssql.SaturnDev, function(err){
    var request = new sql.Request(connection);
    //request.input('input_parameter', sql.int, 10);
    var tst;
    request.input('AppName',sql.VarChar(50),req.query.AppName);
    //request.output(tst, sql.VarChar(50));
    request.execute('SP_MobileAdminGetAppData', function(err, recordsets) {
    // request.execute('procedure_name', function(err, recordsets, return
    // ... error checks
      if(err)
        console.dir(err);
      console.dir(recordsets);
      res.send(recordsets);
    });
  });
```
### Applications attacks - summary

<table>
<thead>
<tr>
<th>Attack</th>
<th>Method</th>
<th>Example</th>
</tr>
</thead>
</table>
| **injections** | - Entering input of SQL language into       | - `select * from users where mail='john.doe@something.com' or 1=1 -- and pass='1234'
|                | vulnerable input areas in a database driven | - `www.mysite.com/viewcontent.php?filename=my_great_content.txt;ls`       |
|                | application.                                |                                                                         |
| **Web**        | Web manipulation                            |                                                                         |
|                | - ClickJacking                              | - `<iframe src="/files/tutorial/window/clicktarget.html"></iframe>`     |
|                | - XSS                                       |                                                                         |
|                | - CSRF                                      |                                                                         |
| **Web services**| SQL Injection.                              |                                                                         |
|                | - XSS.                                      |                                                                         |
|                | - DoS.                                      |                                                                         |
Questions?