Abstract:

The paper introduces Software as a Service which is the latest business and economic model for software delivery. In this model, software is accessible on Internet and available on demand. This is a radical model that has come a long way from traditional ownership model in terms of licensing and sustenance for the end consumer. One of the major challenges that are preventing enterprises and consumers to adopt this model is the threat of Security associated. The paper describes how the principles of software testing can be applied to build confidence around the SaaS model with a proposed framework.
Introduction

We have seen the existence of Software as a Service (SaaS) for many years now. In fact, software applications / products have been marketed as ‘on demand’ business model. Examples for SaaS are Gmail or the more recent one: MS Office 365. SaaS in simple terms can be defined as a business and delivery model where software applications are accessible over Internet and offered as subscription on demand.

SaaS has also evolved from the ASP (Application Service Provider) hosting model where ASPs purchased licensed software from their vendor and provided the software on subscription to the end clients. The model further evolved, marking downfall for ASPs, wherein software vendors deployed and provided the software offerings on subscription basis to end clients. In turn, the end clients (IT enterprises) further provided the software offerings to their own end users.

SaaS is further evolving in terms of hosting. Vendors are moving their on-premise hosting to Cloud hosting. Vendors are also coming up with refined licensing models.

SaaS business model is different from traditional models. In the traditional software delivery model, a customer buys a license for software and assumes ownership for its maintenance as well as upgrades. With the SaaS model there is no need for the customer to pay for the license and maintenance for the software bought which is advantageous to the customer. This ‘Pay per Use’ model offers flexible subscription timelines to end customer.

With so many benefits from the SaaS delivery model, it is expected that enterprises would rapidly move towards SaaS and expect a boost in the ROIs; yet we have seen reluctance in adoption. Many surveys have been conducted for enterprise IT CIOs (Chief Information Officer) and independent users on SaaS adoption where the major concern for reluctance is the challenge of security.

As the software is accessible over the web, there are major security challenges which is discussed in the next section.

SaaS Security Challenges

Although, Security challenges for SaaS are similar to the ones for traditional deployments, yet the magnitude and criticality of addressing them is twofold. This is mainly due to access control and compliance within the organization’s boundary in on-premise deployments.

In a SaaS Model, the control over enterprise data is delegated to SaaS vendor and the interactions happen on the web. In some cases, the SaaS vendor is utilizing a public IaaS for deployment of SaaS. In this case, the enterprise data access and control would be further delegated to the IaaS vendor. Loss of direct control and opaque transactions with SaaS vendor causes hesitation in adopting this model. SaaS offering layers are depicted in the below diagram:
1. Identity Thefts and Access Management Issues:
Identity thefts are one of the biggest threats to SaaS Security (for any software deployment). The security challenge depends on the stringency of IAM model used by the SaaS vendor. Enterprises which use external hosting services are concerned about unauthorized access to the data by SaaS vendor employees or other clients. The secure IAM for each SaaS tenant is ensured by:
   • Storing a part of user credentials and access information within the enterprise and storing only part of it with the SaaS vendor
   • Sharing user credentials and access information securely with SaaS vendor using Federation standards
In both cases, securing the identity of enterprise tenant as well as its end users is a critical factor in SaaS offering.

2. Data Security and Integrity:
Enterprise data needs protection and the accountability lies with the SaaS vendor. The aspect of Data Integrity should be ensured by the SaaS vendor which signifies the fact that data of each enterprise tenant should not be available for other clients throughout the lifecycle of the data. An encryption algorithm and multifactor authentication is used to ensure ‘A level’ data security.

3. Network Security:
All SaaS offering interactions happen over the Internet. SaaS vendor as well as the enterprise should ensure that the sensitive data as well as IAM information is not leaked. Most SaaS vendors nowadays use SSL/TLS, and follow strong network security protocols. Enterprise may connect to SaaS VPN to ensure further security from network interactions point of view.

4. SaaS Compliance to Regulatory Standards:
SaaS offering should conform to vendor policies, government policies and industry standards. Some of these policies can be:
   • EU Data Privacy Law (applicable in many European countries)
   • SOX Act
   • Health Insurance Portability and Accountability Act
   • Payment Card Industry Data Security Standard
   • Gramm-Leach-Bliley Act

5. Physical Security:
This is one of the challenges for physical security of enterprise data and critical information. Any SaaS offering instills greater confidence for adoption if there is a strong disaster recovery in place and thorough failover testing has been done to ensure that there is less risk to client’s data.

Challenges from End User Perspective:
Let us assume that we are using MS Office 365 for writing this paper. The data as well as the hosting is done on Microsoft servers. Let us now list the security issues which an end user can face:
   • The paper could be deleted
   • The contents of the paper could be copied or modified
   • The password or account settings could be changed
   • The audit trails could be changed and the earlier versions could be deleted
   • IP infringement could happen from the enterprise side which can result in the transfer of sensitive information to the MS server
   • MS Server could be hosted in a region that experiences an earthquake and all the data could be lost if proper recovery testing could not be performed
Now that we have highlighted the challenges, let us look at the Security Testing aspect and the issues.
that Security testers face specifically for SaaS Applications. As we know that SaaS interface is customizable for every tenant and hence one time ‘Security Testing’ does not really help in establishing the confidence. The flexibility provided by the SaaS offering involves complex internal logic with layered interfaces. Hence, the security testing needs to be performed at frequent intervals and preferably with each tenant on-board.

**SaaS Security Testing**

Software Security testing is a non-functional testing to ensure that the software under test is securing the sensitive data and related user information while maintaining the data integrity and providing the intended functionality. Security testing should ensure that application is providing:

- Authentication and Authorization
- Availability
- Data Integrity
- Data Security

Following are some of the major security testing performed for SaaS offering. Subsequently, security testing techniques are also discussed for reference.

**Denial of Service:** This happens when the deployment servers or networks are brought down by simulating huge traffic. Hackers use DoS Violation (Denial of Service can be caused through congesting, delaying network packets, or by hacking resources), Man in the Middle Attack (fire brigade attack), IP Spoofing for Denial of Service (DoS) attacks on SaaS. In a multi-tenant environment, the DoS testing have become even more critical as a DoS attack on one tenant (which may utilize 100% CPU or other resources) may cause other tenants to cease working.

**SQL Injection:** Hackers inject SQL queries in input data for getting the sensitive information, change data and may be for crashing the deployment. Functional and Security testing should check that the input data that is being provided to application is filtered for special characters and should verify the easiness to inject queries into the data.

**Hidden Form Parameters:** None of the SaaS application developers should include hidden form fields. Hidden fields introduce security vulnerability for tenants as well as end users and are a big threat to Data integrity as well. White box security testing is the best technique to test hidden form fields.

**Cookie Values:** Security Testing should ensure that data in the cookies is encrypted with strong encryption algorithm and limited sensitive IAM information is being sent out as cookies. Vulnerability scanning is the best technique to perform this testing.

**XSS:** Cross site scripting attacks wherein phishing Web sites are published to the tenants. SaaS are susceptible because they share application access and data among various tenants. Vulnerability scanning and risk based testing can be used to verify whether SaaS offering is susceptible to XSS.

**SaaS Security Guidelines**

Following are some guidelines from DMTF (Distributed Management Task Force) and CSA (Cloud Security Alliance) that should be taken into consideration while planning for Security testing of SaaS deployments:

- When a tenant uses SaaS, the SaaS provider will have to provide a lot of analytical information. This information gathering has to be a regular and a continuous process. Information gathering should be made part of contractual obligation of the SaaS provider
- SaaS providers generally claim security, performance, availability, and other parameters. Claimed security metrics information and acceptability criteria (including legal and compliance requirements) should be included
in any SLAs and pact where applicable. The committed / achieved standards and metrics should be recorded and verified.

- Multi-tenancy and on demand provisioning aspects of Cloud computing add complexity to traditional forms of implementing audit and assessment. So conventional audit and evaluation may be customized in SaaS environment.

- Various security issues, such as suspected data breaches, must be addressed explicitly in specific clauses of the contract that clarifies the respective obligation of the Cloud service provider and the client.

  - Cloud service provider must protect the data from data breach, for e.g., Data that is subject to state and international data protection laws (EU Data Privacy, HIPAA etc) should be protected. The Cloud data provider and data owner must undergo a contract to include penalties to be enforced in case of a data breach.

  - For the hosted applications, Cloud providers must support various security tools for the web applications to provide dynamic analysis on security.

<table>
<thead>
<tr>
<th>Security Testing Technique</th>
<th>Description</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault injection-based testing</td>
<td>Fault injection focuses on the application’s interfaces or boundaries and on the environment in which the application is running. The testing includes verification for all input fields, network interface, file system and environment variables etc.</td>
<td>It determines how the application handles different types of protocol packets. In order to identify security vulnerabilities, fault data should be injected. SDLC stages: Build, unit testing and integration testing.</td>
</tr>
<tr>
<td>Fuzzy Testing</td>
<td>Mechanism of injecting random data into application to determine whether it can run normally under the jumbled input.</td>
<td>Helps in discovering security vulnerabilities. It is a very effective mode of testing and may even find flaws of tested software, which are difficult for the other logical testing method. SDLC stages: Unit Testing, Integration Testing and System Testing</td>
</tr>
<tr>
<td>Vulnerability scanning testing</td>
<td>It includes: Testing space scanning: Running the application to determine leakage that the application might have created. For e.g. network port scanning may identify open ports which should have been closed or not opened at all. Known defects scanning: Running the application to determine problems against a known set of vulnerabilites.</td>
<td>SDLC stages: Unit Testing, Integration Testing and System Testing</td>
</tr>
<tr>
<td>White box testing</td>
<td>Static code review and walkthrough are common tools used here.</td>
<td>Good at finding security bug, such as buffer overflow. SDLC stages: Build: Coding, code review stage.</td>
</tr>
<tr>
<td>Risk based testing</td>
<td>Some research has been done for combining Risk analysis + security testing with software development lifecycle. An early adoption is recommended to find high-risk security vulnerabilities.</td>
<td>SDLC stages: This approach emphasized SDL (Security Development Lifecycle). For e.g.: Creating security abuse use cases (requirements), risk analysis (predesign and design stage), static analysis tools (code), and penetration testing (testing).</td>
</tr>
</tbody>
</table>

Table 1: Security Testing Techniques

17 August 2011: MS Office 365 (SaaS offering) suffered first outage in August. Service promises 99.9% availability.
Based on the security recommendations from CSA, DMTF and NSIT and Security testing results, the following framework can be conceptualized for SaaS Security Evaluation. Based on security testing results, a holistic benchmarking for the SaaS offering can be done. A consolidated report should be published to the SaaS vendor and security recommendations should be provided where the scoring is low. Sample workflow for the framework for a new SaaS offering is given below:

In a multi-tenant environment it is recommended to monitor all the test-tenants for changes in behavior and performance. Ideally, an applied security attack on one tenant should not affect other tenants. The Security testing framework should evaluate and confirm whether the test security attacks on one tenant result into security vulnerability for another tenant.

**Evaluation Methodology**

\[
IS = C \times Wt
\]

\[
TS = \sum_{i=1}^{n} S \times IS_i
\]

\[
MS = \sum_{i=1}^{n} IS_i
\]

Final Score = \( \frac{TS}{MS} \times 100 \)

<table>
<thead>
<tr>
<th>Criticality</th>
<th>Weightage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>0.2</td>
<td></td>
</tr>
</tbody>
</table>

**Score Mapping Table**

<table>
<thead>
<tr>
<th>Score</th>
<th>Ranking</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;90%</td>
<td>5</td>
<td>Very secure! Secure at all the evaluated parameters.</td>
</tr>
<tr>
<td>75%-90%</td>
<td>4</td>
<td>Mostly secure at most of the modules. Some improvements are possible.</td>
</tr>
<tr>
<td>50%-74%</td>
<td>3</td>
<td>Moderately secure. Needs improvement.</td>
</tr>
<tr>
<td>35%-49%</td>
<td>2</td>
<td>Less secure. Redesigning is proposed.</td>
</tr>
<tr>
<td>&lt;34%</td>
<td>1</td>
<td>Unsecure.</td>
</tr>
</tbody>
</table>
Sample Security Test Results:

<table>
<thead>
<tr>
<th>XSS Test Cases</th>
<th>Date Executed</th>
<th>Criticality</th>
<th>$S$ (1 Passed / 0 Failed)</th>
<th>IS</th>
<th>$S^*$IS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo data</td>
<td>10/19/11 20:00:12</td>
<td>P1</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Client side script injection</td>
<td>10/19/11 20:00:12</td>
<td>P1</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Server side script injection</td>
<td>10/19/11 20:00:12</td>
<td>P2</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>.................</td>
<td>...............</td>
<td>P3</td>
<td>0</td>
<td>0.2</td>
<td>0</td>
</tr>
</tbody>
</table>

Final Score = 53(TS / MS*100), It falls under ranking 3.

Table 2: Testing Result for a Security Testing Cycle

Consolidated Report

A Consolidated Evaluation Report for each type of testing as well as recommendations based on the ranking is proposed to the SaaS Vendor.

<table>
<thead>
<tr>
<th>Security Standard</th>
<th>Date Executed</th>
<th>Ranking (Scale 5)</th>
<th>View Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Security</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLAN Security</td>
<td>10/19/11 20:00:12</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Network-based Intrusion Prevention System</td>
<td>10/19/11 20:00:12</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Anti-virus and Firewall strength</td>
<td>10/19/11 20:00:12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Application Security</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing against SQL injection</td>
<td>10/20/11 20:00:12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Testing against XSS</td>
<td>10/20/11 20:00:12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Testing against Denial of Service</td>
<td>10/19/11 20:00:12</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>........</td>
<td>10/19/11 20:00:12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>User Access Management</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role based Access Testing</td>
<td>10/21/11 20:00:12</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>SSO Testing</td>
<td>10/19/11 20:00:12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Legal Compliance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIPAA Compliance</td>
<td>10/19/11 20:00:12</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Summary: 51%
Conclusion

This paper elucidates SaaS Specific Security Challenges and how contemporary Security testing can ensure that the challenges are met. The paper also discusses the major Security standards as laid down by CSA and DMTF that should be followed by SaaS vendors. Based on the Security challenges and Standards, SaaS Security model is proposed for evaluation of security standard for any SaaS offering.

References

No part of this paper has been directly referred from any copyright material. Some reference sites for basic understanding:
http://www.persistentsys.com/LinkClick.aspx?fileticket=EEasuQd0PS0%3D&sttabid=626
http://community.forrester.com/thread/3897
http://www.cigital.com/papers/download/bsi4-testing.pdf
https://cloudsecurityalliance.org/csaguide.pdf

About the Authors

Jayanti Vemulapati is a Principal Technology Architect at Infosys Engineering, heading cloud and distributed computing practices. She has 16+ years of experience in Leading, Designing, Architecting and Executing software projects and products in Java and C across various Domains. She is abreast with the latest technologies and frameworks in Java / J2EE, Web 2.0, Cloud, Distributed File system, SOA, Internationalization and Multi-core. She can be contacted at Jayanti_Vemulapati@infosys.com

Neha Mehrotra is a Technology Architect at Infosys Engineering. She has more than 8 years of experience in Java and Java EE technologies. She has been involved in development and sustenance of identity and access management products. She is currently involved in research involving Testing framework for Cloud based offerings. She can be contacted at neha_mehrotra@infosys.com

Nitin Dangwal is a Technology Architect at Infosys Engineering, Infosys. He has more than 8 years of industry experience and almost 6 years of experience in Federation and Web Security domain. Over past 6 years, he has been involved in development and sustenance of Federation products. He can be contacted at nitin_dangwal@infosys.com