Applying Software Assurance Concepts to the Cloud

Randall Brooks, Raytheon
John Whited, Raytheon

Abstract. It was once said that the last time one had full control of their software was right before they released it. This is ever more important as organizations move applications and services into a public cloud to support a mobile lifestyle. Clouds have been described as "a safe and secure private cloud," "a semi-trusted partner cloud," or "a Wild West full and open public cloud." It is typically toward the latter in which the industry has been moving. Because of this, developers must understand their attack surface and threat environment to ensure that they have focused on "building security into" their applications.

Software Assurance

Software Assurance (SwA) is defined by the National Information Assurance Glossary, Committee on National Security Systems Instruction (CNSSI) No. 4009, as "the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the software functions in the intended manner." SwA focuses on "building security in" to ensure the software is resilient in harsh operating environments such as those in which cloud applications operate. According to the DHS, SwA addresses:

- **Trustworthiness**
- **Predictable Execution**
- **Conformance**

In a cloud environment, Trustworthiness is key to providing confidence to the user that no exploitable vulnerabilities exist. With use of mobile and the cloud, users will demand Predictable Execution. Cloud applications that function as intended will avoid user frustration. Cloud applications must conform to appropriate secure APIs to ensure interoperability with other applications and services.

To deliver SwA, various principles can be applied such as determining the criticality of the application, defining the Attack Surface, developing misuse cases, and testing for unintended consequences. But these principles have a slightly different emphasis when applied to the cloud.

Determine Component Risk and Criticality

As with other complex problem spaces, understanding Attack Surface, the threat environment, and related elements of risk management in the cloud is best addressed by first decomposing the problem space into its constituent components. Stepping down in the hierarchy from "the cloud," we find the following:

- A cloud environment is composed of interacting systems.
- Each system may be comprised of multiple subsystems.
- Systems and subsystems are built from one or more components – at a software level, the host OS, most likely one or more guest operating systems, and the applications that are controlled by the OS and use OS services.

This decomposition can be extended arbitrarily to the required depth, until assessing the risk of the lowest level element (here, a "component") can be meaningfully addressed. At this lowest level, the risk associated with a component can be characterized by assessing the following:

- Weaknesses or vulnerabilities inherent in the component
- The likelihood that a weakness or vulnerability can be exploited
- The existence of a means to exploit the weakness or vulnerability
- The presence of a threat actor with the skills and access to launch an exploit
- The impact of a realized exploit should it occur

Once a risk profile is created using these factors for each component, a combined risk analysis can then be undertaken on subsystems and then systems while considering the operational importance to the organizational mission. This top-down decomposition, component risk assessment, and bottom-up reconstitution yields a structured means of assessing the risk associated with a complex cloud environment.

Threat Modeling: Define the Attack Surface and Develop Misuse Cases

To further understand component risk, one must understand the specific means by which a threat actor might be able to exploit a vulnerability. This is the component's Attack Surface. Once decomposition is accomplished, a good way to define one's composite Attack Surface is to perform Threat Modeling on each component. Microsoft teaches an application centric method called Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege (STRIDE). Aspects such as Denial of Service (DoS) and Elevation of Privilege have a more pronounced importance in cloud applications.

For example, due to the need of accessibility to data, an effect of a DoS attack may cause far worse consequences in a public cloud than a private cloud. A prolonged DoS attack may also affect user perception of how well an application operates.

Further, if one hosts their cloud application with a cloud provider, it is likely that their application will reside on a virtual machine (VM) running on a large server class system with other VMs, as many public cloud-based systems utilize hosted VMs. In this virtual environment, privilege escalation may have a far-reaching effect on one's hosted application and other applications hosted on the same system. As depicted in Figure 1, a successful privilege elevation attack of a hosted VM may put other hosted VMs in one's "digital neighborhood" at risk.

If the cloud application in "App B" has an exploitable vulnerability, successful privilege elevation to the "Host Operating System", may enable an attacker to mediate or disrupt "App A" or "App C."

Given the example above, threat modeling can help one think about boundaries to applications and the various interfaces that cross these boundaries. This effort can help one derive misuse cases and think about how an attacker could leverage these interfaces and discover flaws in one's design. Requirements and test cases can then be developed to focus on a cloud-based environment. These requirements and associated test cases may highlight bugs in the form of weaknesses as listed in the Common Weakness Enumeration (CWE), or in the form of exploitable vulnerabilities as listed in the Common Vulnerabilities and Exposures. These requirements can be verified and validated over time with periodic static and dynamic testing.

Static and Dynamic Testing

According to the Cloud Security Alliance (CSA) Research group, the top cloud threats are "Trust, Data Leakage, Insecure Cloud software, Malicious use of Cloud services, Account/Service Hijacking, Malicious Insiders, and other Cloud-specific attacks." Testing one's cloud application prior to deployment and early within the System Development Lifecycle is important to ensure SwA of the cloud application.

Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) are critical activities to ensuring secure coding practices are verified and that no latent vulnerability remains. For DAST, additional focus should be applied to simulating a cloud environment and to fuzz testing input and protocols.

SAST should focus on the CWEs most relevant to one's mission. With knowledge of one's component risk and criticality, the Common Weakness Risk Analysis Framework can be applied to prioritize weaknesses. For example a "Use of a One-Way Hash without a Salt (CWE-759)" may be rated a lower effective risk than "Double Free (CWE-415)." A Double Free can create corruption that causes the program to crash leading to service interruption.

DAST should focus on exercising and testing the Attack Surface that was identified during threat modeling. Fuzz testing of all inputs may reveal coding weaknesses that represent exploitable vulnerabilities. For example, consider a "Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') (CWE-120)" that exists within the application server. A malicious distributed client could be deployed by a malevolent actor to try multiple concurrent input attempts. Each input provides data which is accepted by the system, but which exceeds the expected size of the buffer receiving the input. The extraneous data can be crafted to give control of the application to the attacker.

DAST is particularly useful for testing backend systems such as Databases. One of the biggest risks to a cloud application is

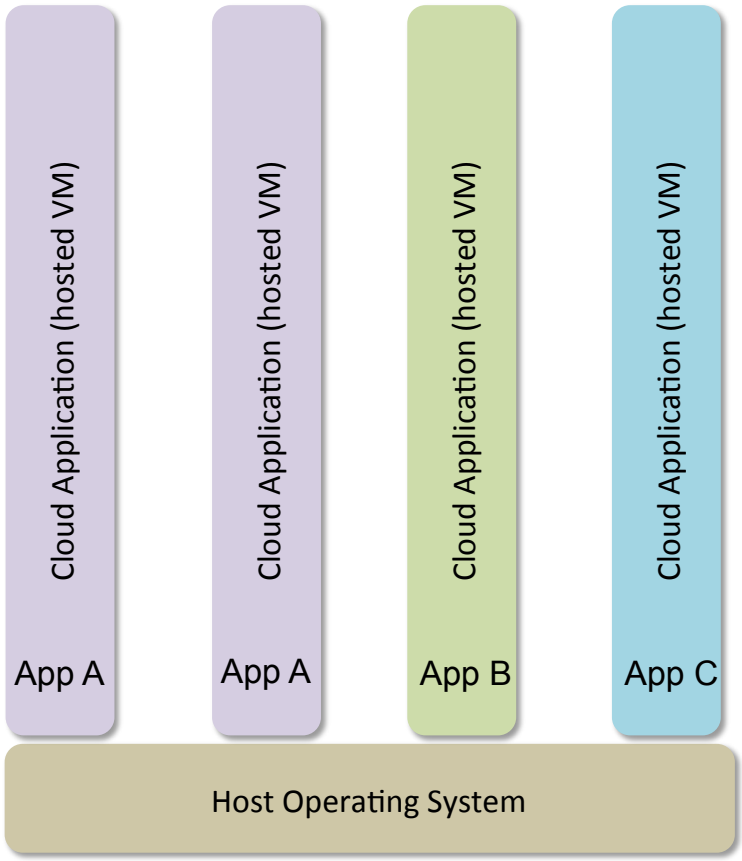

Figure 1. Digital Neighborhood

unsanitized input that is treated as a valid database SQL command. The Open Web Application Security Project (OWASP) number one "Application Security Risk" is entitled "A1 – Injection". OWASP states "Injection flaws, such as SQL, OS, and Lightweight Directory Access Protocol injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data." CWE/System Administration, Networking, and Security Institute Top 25 Most Dangerous Software Errors notes this as Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') (CWE-89). DAST can be leveraged to dynamically execute SQL commands injected into the backend database. This helps to locate non-parameterized queries prior to production release.

Since the kinds of applications under discussion will be in a threat environment that is fully open, 3rd party testing is more critical than ever. Internal development houses may have their "blinders" on and may lack the "Think Evil" gene required to appropriately test their application. A coder's ability to understand how one compromise might be leveraged to obtain something of greater value may be lacking. Although education of software writers over time is important, this shortcoming in software staff is currently prevalent and suggests that 3rd party testing is a key to successful application security testing in a cloud environment.

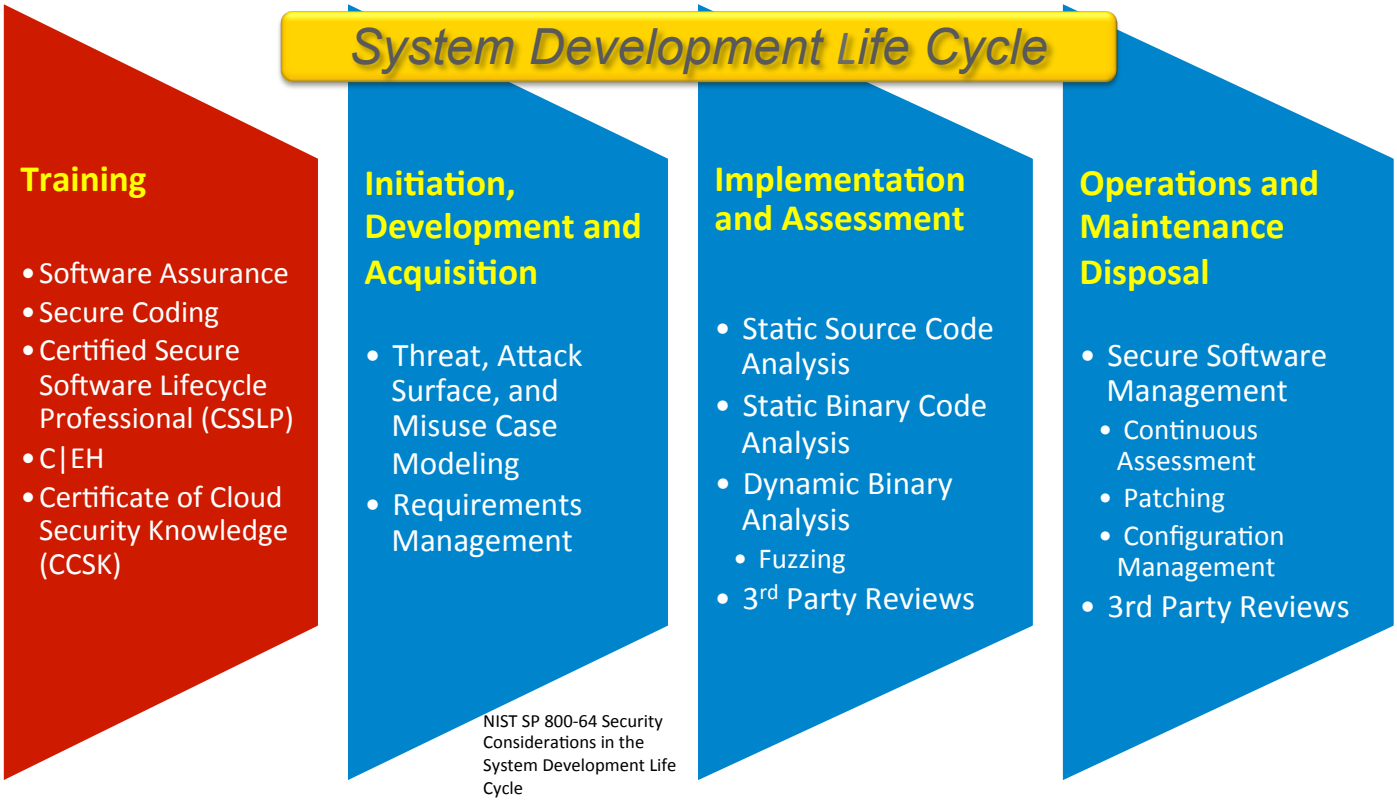

Figure 2. Software Assurance for the Cloud

Additional Items to Consider

As shown in Figure 2, the NIST Special Publication (SP) 800-64 Revision 2 entitled "Security Considerations in the System Development Life Cycle" covers the Initiation, Development/Acquisition, Implementation/Assessment, Operations and Maintenance, and Disposal phases of a development Life Cycle. The topics previously discussed within this article align with these phases, but can be further augmented with training:

- **Software Assurance – Training covering a general overview of SwA and its importance**
- **Secure Coding – Training covering secure coding practices to avoid common programming mistakes**
- **Certified Secure Software Lifecycle Professional (CSSLP) – A professional certification by the International Information Systems Security Certification Consortium (ISC)², which includes eight Common Body of Knowledge domains**
- **Certified Ethical Hacker (C|EH) – A professional certification provided by the International Council of E-Commerce Consultants (EC-Council)**
- **Certificate of Cloud Security Knowledge (CCSK) – A professional certification by the Cloud Security Alliance which covers 14 domains across 3 sections (Cloud Architecture, Governing in the Cloud, and Operating in the Cloud)**

Although not required when adequate 3rd party security test staff is available, training of both systems and software staff is important as it helps to address issues early in the Life Cycle.

On the other end of the Life Cycle is Operations and Maintenance. One must consider the importance of availability in a cloud application. Patching vulnerabilities identified may have to take place while that system is online and operating, which may lead to data integrity and stability issues. In private clouds it may be permissible to schedule a maintenance window, and turn off portions of cloud application functionality. In the highly dynamic public cloud, where resilience is key, patching and operating in a redundant fashion will enable smoother transitions to upgraded versions.

With specific reference to cloud applications, the CSA publishes the Governance, Risk Management and Compliance (GRC) stack at <https://cloudsecurityalliance.org/research/grc-stack>. The GRC stack covers a suite of four integrated initiatives: the Cloud Audit, the Cloud Controls Matrix, the Consensus Assessments Initiative, and the Cloud Trust Protocol. By applying SwA concepts to one's cloud application built in compliance with the GRC stack, one's application should be far more resilient to a harsh cloud environment.

Conclusion

It is important that organizations understand the criticality of their application or service with respect to its organizational value. With this knowledge, one needs to understand their Attack Surface and what affect a threat may have to their critical program information. Performing Threat Modeling and focusing on architectures can help one derive testing requirements. Internal and external SAST and DAST testing can go a long way to ensure SwA of their cloud application. Always remember that the time spent testing the application internally for security flaws will be dwarfed by the time a determined attacker may be willing to spend attempting to exploit that application.

ABOUT THE AUTHORS

Randall Brooks, Engineering Fellow, Raytheon, has more than 15 years of experience in Cybersecurity with expertise in Software Assurance (SwA) and secure development life cycles (SDLC). He has been awarded three US patents on Intrusion Detection and Prevention, and three US and one UK patent(s) on Cross Domain solutions. He is also a CISSP, CSSLP, ISSEP, ISSAP and an ISSMP. He is a graduate of Purdue University with a Bachelors of Science from the School of Computer Science. He represents Raytheon within the U.S. International Committee for Information Technology Standards Cyber Security 1 (CS1).

Phone: 727-302-2189
E-mail: brooks@raytheon.com

John Whited, Principal Engineer, Raytheon, has 5 years of experience in Cybersecurity with expertise in Software Assurance (SwA) and secure development life cycles (SDLC). Prior to joining Raytheon, he was a software and a systems engineer in commercial telephony, holding five US patents on Intelligent Networks. He is also a CISSP and a CSSLP. He is a graduate of Texas Tech University with a Bachelors of Science and a Masters of Science in Electrical Engineering. He has made two joint presentations at the RSA Security Conference (2010 and 2012).

Phone: 972-205-4750
E-mail: john.whited@raytheon.com