Q&A

Sherif Koussa

Q. Should startups care about application security?

A. Many of the startup executives I meet think that application security is only for large companies, such as banks and government agencies. After all, these organizations have a lot of data to secure, established reputations to worry about, and trusted brands to protect. Startups do not have those things (yet), nor do they have the money to invest in anything that will not help them reach the next round of financing. They are focused on acquiring customers to establish better brand recognition. So, it is easy to understand why startups may be less than enthusiastic about the topic of application security. However, in my experience, many smart and successful CEOs and CTOs are not so quick to dismiss the topic. Startups that do not pay enough attention to security in the early stages may fail to later capitalize on the value of what they are building now. Furthermore, successful startup executives recognize the value of security as a market differentiator.

Unfortunately, it is not enough for startups to recognize that they need to care about application security; they need to take action. The challenge is cutting through the apparent complexity and building-in application security from the very beginning, while minimizing costs. Here, I will provide an overview of the key elements of application security, and I will discuss practical strategies that startups can use to increase the security of their applications throughout their lifecycles. I will focus on how a startup team can protect its application code against malicious threats, although a startup should also consider how security can provide opportunities to differentiate its application from the competition create opportunities in the marketplace.

Security Design and Architecture

When designing a secure application, security design and architecture is the key. There are six security cornerstones that must be kept in mind in every stage of the design:

1. **Confidentiality:** limiting access to data to only those who should have access to that data.
2. **Integrity:** ensuring that data has not been modified either accidentally or maliciously, either in transit or at rest.
3. **Availability:** ensuring that the data and the systems serving this data are up and running when needed.
4. **Authentication:** confirming the identity of a user or a system and proving that they actually are who they claim to be.
5. **Authorization:** ensuring that the authenticated entity has access rights to the resources that they claim they have access rights to.
6. **Non-repudiation:** proving whether or not an entity actually made a transaction they claim to have made.

Once these cornerstones have been established, there are three design concepts that come to play: attack resilience, attack tolerance, and attack resistance.

1. **Resistance:** the ability of the software to resist attacks. Principles that help with attack resistance include:

- *Defence in depth* (tinyurl.com/m5lkblc): building the security of a system in layers such that result is greater than the sum of its individual parts.
- *Attack surface* (tinyurl.com/5py7w4): minimizing the attack surface, which is those places where an attacker can start poking the application looking for holes.
- *Least privilege* (tinyurl.com/29a93a): giving users and processes only the minimum set of privileges to perform their function.

2. **Tolerance:** the ability of the software to tolerate failures. A principle that helps with attack tolerance is *failing securely* (tinyurl.com/h7vhm): a very important design principle that entails anticipating and handling exceptions in the software, so that the software does not end up in an insecure state in a fail scenario.

Q&A. Should Startups Care about Application Security?

Sherif Koussa

3. **Resilience:** the ability of the software to isolate attacks and contain the damage resulting from these attacks. A principle that helps with attack resilience is *compartmentalization*: an object-oriented programming concept that entails segregating different modules of the software. If a module is breached, it may be contained within that module and not necessarily spread to the whole application.

Increasing Awareness and Knowledge

The general strategies described above can help a startup improve its security posture, but they may still seem difficult to implement. However, it does not have to happen all at once. Software is developed in phases, and software security is built in the same way. The key is to take small yet measurable and progressive steps towards the goal. In many cases, the first step is for the startup to increase its staff's awareness and knowledge of security issues.

Companies should review their application-security awareness and security design. Even just knowing that an issue exists or is important can help a startup manage the associated risk. Ira Winkler (2012; tinyurl.com/acuofmc) argues that security awareness can be the most cost-effective security measure. Many code flaws happen because developers lack knowledge about proper secure coding and the reasons and consequences of writing a certain line of code in a certain way.

A great place to start increasing awareness and knowledge is by taking courses, either in person or online. I have also seen companies do very well with "lunch and learns" or similar in-house seminars with experts. Startup teams can also review lists of common security flaws, such as the "Top 10 list" published by the Open Web Application Security Project (tinyurl.com/3n6q9rg), both to increase awareness and assess their own application's security. Deliberately insecure applications in various languages are also available for testing and learning purposes; an example is WebGoat (tinyurl.com/62kggay) for Java-based web applications.

Taking Action Through Controls

Once security awareness has been established, safeguards or countermeasures must be put in place to ensure that the knowledge obtained during the awareness phase is actually implemented in the code. Usually, application-security controls are divided between preventative and detective controls:

1. **Preventative controls:** These controls include the security awareness implemented in the previous phase and other controls. Examples include:

- *Security checklists*: Checklists are the most effective security controls, yet their value is frequently underestimated and they are underused. A security checklist is simply a list of all the things a developer should check before committing code to the repository. Helpful resources include Mozilla's Secure Coding Guidelines (tinyurl.com/4ynfbqn) and Secure Coding QA Checklist (tinyurl.com/km3et2m), as well as MSDN's Secure Coding Guidelines (tinyurl.com/67a6ne9). Also, Patch++ (patchplusplus.com) provides a visual way to implement checklists for securing code patches.

- *Security code review*: There are many flavours of security code review, but the simplest form is a regular peer code review infused with security guidelines and checks developed from the security checklists mentioned above. The most inclusive form is a full-scale security code review involving the use of automated tools and scripts as well as manual inspection of the code. Security code review is one of the best controls for a software development lifecycle; it can prevent the largest number of security flaws from making it to production, and it provide the quickest means of remediation. For my simplified version of a security code review process, see Koussa (2013; tinyurl.com/kbhwy3s).

2. **Detective Controls**: The two most-common detective controls in application security are:

- *Penetration testing*: with this control, an internal or external security analyst tries to emulate what an attacker would do to look for vulnerabilities in a given piece of software and then tries to exploit them. Other names for this type of control include vulnerability assessment, vulnerability scanning, or dynamic testing, but they all represent more or less the same type of control with different levels of thoroughness. Penetration testing is a very good control to measure the "hackability" of the application.

- *Web-application firewalls*: these are firewalls that monitor traffic going in and out of a web application. Depending on the firewall's capabilities, the firewall could potentially block inputs and outputs that do not meet the criteria defined in its set of rules. Web-application firewalls are often a popular option to protect deprecated or soon-to-be-deprecated applications. They are also a popular choice to provide some protection for applications that are deemed too costly to fix.

Q&A. Should Startups Care about Application Security?

Sherif Koussa

Implementing Processes

Once a startup team is aware of the security threats, risks, and attacks that are relevant to their application, once they knows what needs to be done in order to counter these attacks, and once they are implementing a few controls to ensure that the awareness is practically implemented, the next phase is to implement a systematic and measurable process across all disciplines of the software development lifecycle to ensure that fewer and fewer vulnerabilities make it to production. There are several approaches to securing software-development lifecycles, such as Microsoft SDL (tinyurl.com/y6frgge), or initiatives that help integrate security into existing models, such as BSIMM (bsimm.com) and OpenSAMM (opensamm.org).

The challenge for any process is whether it actually is adopted by the development teams, who may not welcome adding additional processes if they perceive process to interfere with the actual job of writing code (Turner, 2011; tinyurl.com/44xh5sw). When it comes to choosing a secure software development lifecycle process or introducing new security activities into existing ones, I always suggest small yet progressive steps. Nothing is more damaging than to shock development teams by suddenly imposing heavy processes.

Conclusion

Startups can no longer afford to ignore application security. It is not a question of whether or not startups should care about application security; they need to do more than care – they need to take action. However, taking effective steps toward secure software does not have to come with a hefty drain on the startup's budget or productivity levels. On the contrary, some startups are using software security as a marketing differentiator in an age when clients are looking for more privacy and demanding evidence of privacy controls implemented by the organization.

Recommended Reading

- "Application Security Architecture" (Simhadri, 2001; tinyurl.com/nyu7lzc)
- Software Security Engineering: A Guide for Project Managers (Allen et al., 2008; tinyurl.com/lua92tb)
- "Architecture and Design Considerations for Secure Software" (SwA Forum and Working Groups, 2012; tinyurl.com/mmx928h)

About the Author

Sherif Koussa is Principal Application Security Consultant and founder of Software Secured, an application security firm. He has spent 14 years in the software development industry, with the last six years focused on testing application security, assessing security, and teaching developers to write secure code. He worked on the OWASP security teaching tool WebGoat 5.0, helped SANS launch their GSSP-JAVA and GSSP-NET programs, and wrote the blueprints of the Dev-544 and Dev-541 courses. In addition, he authored courseware for SANS SEC-540: VOIP Security. Sherif leads both the OWASP Ottawa Chapter and the Static Analysis Code Evaluation Criteria for WASC. He has performed security code reviews for three of the five largest banks in the United States. Before starting Software Secured, Sherif worked on architecting, designing, implementing, and leading large-scale software projects for Fortune 500 companies, including United Technologies, and other leading organizations such as Nortel Networks, March Healthcare, Carrier, Otis Elevators, and NEC Unified Communications.

Citation: Koussa, S. 2013. Q&A. Should Startups Care about Application Security? *Technology Innovation Management Review*. July 2013: 50–52.

Keywords: cybersecurity, application security, software security, training, checklists, startups, code reviews, detection, prevention, design, architecture