OneSpan
Be bold. Be secure.

RUNTIME APPLICATION SELF-PROTECTION IS CRITICAL FOR MOBILE APP SECURITY

WHITE PAPER

TABLE OF CONTENTS

About the Author

David Strom is one of the leading experts on network and Internet technologies.

David Strom (@dstrom, strominator.com) is one of the leading experts on network and Internet technologies and has written and spoken extensively on topics such as VOIP, convergence, email, cloud computing, network security, Internet applications, wireless and Web services for more than 25 years. He has had several editorial management positions for both print and online properties in the enthusiast, gaming, IT, network, channel, and electronics industries, including the editor-in-chief of Network Computing print, DigitalLanding.com, and Tom's Hardware.com. He began his career working in varying roles in end-user computing in the IT industry. He has a Masters of Science, Operations Research degree from Stanford University, and a BS from Union College.

INTRODUCTION

Today most of us go about implementing security from the outside in. The common practice is to start by defining a perimeter and trying to defend it with various security tools. Even though perimeters have been porous for more than a decade, we still can't give up this notion that if we build a better wall we can keep our enterprises safer.

Certainly that is where most enterprises are spending their security budgets. Gartner [1] estimates we spend more than 20 times as much protecting the perimeter than on the actual apps that we know and love and run every day. (The actual figures are $11.5 billion compared to half a billion.) But that kind of investment needs to end. It is time to start thinking about our enterprise security from the inside out, and start protecting our most precious IT assets, our apps.

Apps are hard to protect, to be sure. They are on devices that aren't always owned by the enterprise, and these devices operate outside our corporate perimeter protection. The operating systems are easily out of date and many apps on them have bugs: Gartner estimates a majority of apps have programming errors that can be easily exploited by hackers.

The world has become more mobile

One reason why apps are so insecure has to do with the fact that business users consume more mobile apps now than ever before. And as more computing happens on mobile devices, the risk goes up that they could be compromised.

A second reason is that as more businesses write and deploy their own mobile apps, they increase the potential attack surface and make a corporation more vulnerable than ever to attack. Hackers know this and are using the same social engineering techniques they honed with phishing attacks to make compromised fakes of these apps and get users to download them, allowing them access to these mobile devices. But even without these fakes, mobiles are more vulnerable to ordinary phishing emails. This is because busy users are more likely scanning their mobile inboxes and could be more likely to click on phished email links. That one phished email can create all sorts of havoc across an enterprise.

Mobile devices are also a challenge to maintain. Keeping a mobile device updated with the latest OS and security patches is a lot more difficult than maintaining desktops, especially when they are in the hands of customers and not employees. And in some markets, they are jailbroken or rooted intentionally by end users, making their security even more suspect and difficult to implement, since once rooted, applications can pollute the entire device and steal data.

Another issue is that all apps aren't equal, but all can be compromised. As examples:

- Some third-party apps written by the usual enterprise software vendors carried a fake certificate issued for a subset of Windows Live users by Microsoft [2] last year,
- Your own development team might have inadvertently included flaws that create security holes,
- Or the numerous kinds of code injections and other programming compromises that can be done on apps [3],
- Or your users have mistakenly or ignorantly downloaded infected apps from the Apple iTunes and Google Play stores.

This last point bears some additional explanation. Sadly, the app stores are riddled with fake or infected apps galore. Checkpoint disclosed that recently discovered malware-infected apps, some which have been available on the app stores since April, have been downloaded from the Google Play store as many as 2.5 million times [4]. And even Apple's iTunes Store isn't completely safe: a story from Tech News World in September [5] has documented at least 800 infected apps there, some of which have millions of downloads.

Keeping up with these exploits isn't easy. As the number of mobile apps in use continues to grow, "Technologies and services that we use to test and diagnose our applications for security vulnerabilities fail to scale to test all applications and to test them with the necessary accuracy. There are too many apps, testing skills are scarce, and tools are too complex and inaccurate." [1]

On top of these challenges, there are a variety of mobile security standards that make it hard to enforce best app security practices. Each mobile device vendor is more focused on ease of use rather than built-in security.

Adding this all up, having mobile apps means managing more risk. "When one considers the huge number of applications run by major institutions, the third-party libraries included, and the frequent updates to those applications, it becomes clear that application security is an extremely challenging task indeed." [6]

Malware is getting more complex

In addition to the difficulty of securing mobile apps, malware is getting more complex and sophisticated at finding weak spots in the enterprise infrastructure. It is more than just creating fake mobile apps, but leveraging the actual program code that every app uses.

Malware used to be more easily detected through residues of files or simple signatures that were an obvious sign of infection. Those days are sadly gone. Modern malware is more insidious, and can gather small bits of code that is already written. Using techniques such as return oriented programming, malware can execute standard libraries and other executable sequences of code that can compromise an otherwise uninfected system.

This means that apps themselves – even ones that have been carefully crafted – can be a threat.

> **"... apps themselves – even ones that have been carefully crafted – can be a threat."**

The rise of agile development cycles

There is one final wrinkle in the story to secure our apps, and that is the nature of how these apps are being constructed has changed radically in the last several years. Gone are the days where an IT shop would take months to refine requirements, build and test prototypes and deliver a finished product to an end user department. The idea almost seems quaint nowadays. Instead, we have new working methods, called continuous deployment and integration that refine an app daily, in some cases hourly. Feedback is nearly constant, which means changes are too.

This presents a problem in protecting these apps. "Security products must do more than address application security issues; they need to mesh with continuous integration and continuous deployment approaches, while offering automated capabilities and better integration with developer tools." [7]

Enter the world of RASP

To solve some of these issues with insecure apps, a new kind of app protection is required. As we said earlier, we have to start protection from inside of the apps themselves, and add security that can be part of the app's actual source code. This is the concept of runtime application self-protection or RASP. The idea is slowly catching on. The term comes from a 2012 Gartner report [8] and has several supporting tools vendors.

RASP sounds like a good idea: One blog post on SearchSecurity has put it this way: "It is every app for itself out there, and having firewall-like capabilities built into runtime environments may turn out to make a lot of sense." [9] And Gartner emphasizes this: "There are two capabilities that should be built into any application's runtime — self-protection and self-testing/self-diagnostics." [1]

These RASP products cover a variety of circumstances, use cases and protective methods. There is no single deployment model in common, and many RASP products support a wide variety of programming languages and integration methods, along with different detection and prevention features.

There are two basic RASP implementation methods [8]: One way is for the RASP products to use agents that monitor the apps. This doesn't require any code changes to the apps themselves. Another method is to use some kind of code libraries or build protection in a software development kit, which has to be integrated into the application. This could take the form of replacing existing code libraries, with ones that are RASP-enabled. These tools will require recompiling the apps to include the RASP libraries.

RASP's protection measures include one or more of the following elements:

- The ability to terminate user sessions
- The ability to terminate an app, without affecting other apps on the server or device, or at least identify a misbehaving or compromised app
- When an app is compromised, an alert is sent to a management console or to the user directly
- The ability to inspect application logic flow and data flow
- The ability to connect to the application runtime processes and environment

Typically, the RASP tool supports a variety of programming environments, such as Java, Objective C and Swift along with most common development frameworks. Some tools are just designed for desktop apps, while others can handle apps running across a wider range of desktop and mobile OS's. This is also a challenge for RASP products: the sheer diversity of languages, frameworks, web-facing interfaces and data types means that these products have to support a lot of different circumstances, use cases and environments.

Some RASP vendors offer specific features that are mapped to threats (such as a feature that detects and blocks the SSL Heartbleed compromise). This helps the security team show particular compliance and can make a RASP product more appealing to management. Leading providers offer a more comprehensive protection scheme. Here is an example of the kinds of intrusions that are either prevented or detected by a strong RASP solution:

What intrusions are Prevented or Detected?

INTRUSION	Android	iOS
Debugging	Prevent Java & Native Debugger	Prevent Debugger
Privilege Escalation	Detect Root	Detect Jailbreak
Emulator	Detect Emulator	
Code modification	Detect Java Hooking Framework & Native Code Hooking	Prevent Runtime Library Injection and execution
Application impersonation	Detect repackaging	Detect repackaging
Data leakage	Prevent user and system screenshot Detect untrusted Keyboard	Prevent system screenshot Detect user screenshot Prevent Keyboard cache

As you can see, RASP protection has a lot of depth and can manage the increased risks of rising mobile app use and more sophisticated malware in today's enterprises. They can be a proactive means of protection that other security tools can't easily deliver.

The cost/benefits of RASP

The benefits of RASP are clear: make your app portfolio more secure, and your apps more reliable. But once you get beyond this platitude, it is a lot more difficult to articulate why you should invest in RASP. For example, if you have a lot of internal app development, it might be more difficult to justify the costs.

This is because protection has to be applied to each app, and has to be included as part of the devops process. There could be performance impacts on the apps that use RASP. It isn't a magic bullet: if you write substandard code, it will still be substandard code with RASP.

One issue for RASP is that it bridges the divide between the infrastructure and app server owners – typically these are IT and an end user department, respectively. Another is that you are injecting code into production apps, which may be a political hot potato; depending on how long these apps have been in production and what effect the RASP code has on the production app itself.

In short, there are a number of considerations around how best to secure applications with RASP to ensure a good fit with the development process and cost/benefit. However, there remains no question that natively integrated RASP technology is a very compelling solution that helps address the overall strength of mobile applications.

The rise of defense in depth

RASP shouldn't be your sole security solution, and most experts recommend it is just one part – albeit an important one – from an overall defense in depth and using a series of layers to protect your enterprise. But RASP can be complementary, rather than compete, with these other layers, because it operates at the app level and protects individual apps. Other protective products, such as web application firewalls and mobile device managers, don't have the granularity of protection and can't get down and dirty with the code used to construct apps.

RASP tools can also handle automated setup and can learn app behavior to examine what is defective in terms of overall security. As Securosis [6] has said, "There is absolutely no reason you can't run RASP alongside your existing WAF. RASP solutions are much more effective at attack detection than web application firewalls because they actually see what's really happening, and can more effectively apply security controls." Neil MacDonald, a Gartner analyst, feels that WAF and RASP are complementary, and have great synergy. "The trick with either technology is realizing that you can't find every potential exploit or vulnerability when you develop your code." [11]

Enter the OneSpan Mobile Security Suite

OneSpan's RASP solution is part of its Mobile Security Suite. It contains features such as jailbreak or root detection, along with the ability to determine if an app has been tampered or otherwise repackaged or code injected into the app programming process. It can also determine if execution flow of the app has been modified and check for the integrity of various app processes. Finally, OneSpan's RASP solution can determine if keyloggers or debuggers are being used and if the app is being run inside an emulator.

RASP is just one of numerous other protection features available in the OneSpan Mobile Security: there are other tools such as next generation biometric authentication methods (i.e. Face recognition), sophisticated risk based authentication and others. Additionally, the product offers end-to-end encryption to completely protect communication channels.

Summary

Yes, it is an insecure world out there, and today's app portfolio is responsible for the numerous vulnerabilities businesses face. But RASP can be a solid defense and a way to isolate and neutralize a potential threat, so you can operate your business safely in these uncertain environments. OneSpan's Mobile Security Suite with RASP technology offers a comprehensive approach to app security and critical tools supporting the future of app development and security.

[1] Gartner, September 2014, Maverick Research: Stop Protecting Your Apps; It's Time for Apps to Protect Themselves www.gartner.com/doc/2856020/maverick-research-stop-protecting-apps

[2] Computer Weekly: www.computerweekly.com/news/2240242434/Microsoft-warns-of-fake-SSL-certificate-for-Windows-Live

[3] AppDevelopment Magazine: appdevelopermagazine.com/4398/2016/9/20/Think-Your-Mobile-App-is-Hack-Proof?-Think-Again

[4] ArsTechnica: arstechnica.com/security/2016/09/two-critical-bugs-and-more-malicious-apps-make-for-a-bad-week-for-android/

[5] Tech News World: www.technewsworld.com/story/82521.html

[6] Waratek, April 2015, RASP: Making Apps Self Protecting, Self Diagnosing and Self Testing docplayer.net/14422951-Runtime-application-self-protection-rasp-making-applications-self-protecting-self-diagnosing-and-self-testing.html

[7] Securosis, August 2016, Understanding and Selecting Runtime Application Security & Protection: www.businesswire.com/news/home/20160823005315/en/Securosis-Identifies-Benefits-Runtime-Application-Self-Protection-RASP#.V73pq22c-Lg.email

[8] Gartner, September 2012, Runtime Application Self-Protection: A Must-Have, Emerging Security Technology: https://www.gartner.com/doc/1994816/runtime-application-selfprotection-musthave-emerging

[9] SearchSecurity, March 2015: Is runtime application self-protection a shortcut to secure software? searchsecurity.techtarget.com/opinion/Is-runtime-application-self-protection-a-shortcut-to-secure-software

[10] Gartner, April 2015, Six Principles of Mobile App Security Testing: www.gartner.com/doc/3030823/principles-mobile-app-security-testing

[11] Private interview, September 2016

OneSpan

OneSpan enables financial institutions and other organizations to succeed by making bold advances in their digital transformation. We do this by establishing trust in people's identities, the devices they use, and the transactions that shape their lives. We believe that this is the foundation of enhanced business enablement and growth. More than 10,000 customers, including over half of the top 100 global banks, rely on OneSpan solutions to protect their most important relationships and business processes. From digital onboarding to fraud mitigation to workflow management, OneSpan's unified, open platform reduces costs, accelerates customer acquisition, and increases customer satisfaction.

CONTACT US

For more information:
info@OneSpan.com
www.OneSpan.com