THE DZONE GUIDE TO

APPLICATION AND DATA SECURITY

VOLUME II

BROUGHT TO YOU IN PARTNERSHIP WITH

bmc

NGINX

SYNOPSYS®

WhiteHat SECURITY®

Dear Reader,

Information Security; hardly a day goes by without media coverage of an intrusion or attack. 2016 has seen significant growth in Internet usage. Unfortunately, along with this sudden explosion of connectivity, comes increased opportunity for criminal activities. We now face well-organised criminal gangs and political "hacktivists" on a scale that has not been seen before. Online technology has become weaponized, and as a result, we have created our own Frankenstein's monster.

With all of the security threats that have arisen in 2016, are we now at a saturation point, or is 2017 going to bring yet more challenges to an industry that needs to respond and adapt faster than any other? At the core of a working global Internet is Information security, and without it we couldn't all occupy such a rich infrastructure. However, fragility comes with this complexity and size. The question that remains – is disruption always a bad thing?

Traditional Internet security has typically been an action-and-response environment where an exploit was used, identified, and then addressed. With more people using mobile devices than ever before and an unprecedented explosion in IoT devices, what does it mean for us in the industry?

In this year's *DZone Guide to Application and Data Security*, our industry experts will examine key aspects of the changing landscape of global Internet security, how we can address today's threats, what we can expect from tomorrow, and if we have ignored lessons from the past.

Our expert contributors will cover all aspects of the digital threat, covering server, infrastructure, and application security along with what tools, systems, and people we have to help combat the threat to our digital way of life. There is no point in continually fire-fighting a problem without getting to know a little about the sources, so we will attempt to introduce you into the mindset of those intent on subverting security with a view to understanding how we can combat them.

Welcome to the new DZone *Guide to Application and Data Security for 2016*. Here at DZone we are extremely excited and we cannot wait for your contribution and feedback. Did our industry experts get it right? Join in with the conversation and let us know!

Hey, let's be careful out there.

BY PAUL LUSH
DIRECTOR OF SYSTEMS OPERATIONS, DZONE
RESEARCH@DZONE.COM

TABLE OF CONTENTS

EDITORIAL

CAITLIN CANDELMO
DIRECTOR OF CONTENT + COMMUNITY

MATT WERNER
CONTENT + COMMUNITY MANAGER

MICHAEL THARRINGTON
CONTENT + COMMUNITY MANAGER

NICOLE WOLFE
CONTENT COORDINATOR

MIKE GATES
CONTENT COORDINATOR

SARAH DAVIS
CONTENT COORDINATOR

TOM SMITH
RESEARCH ANALYST

BUSINESS

RICK ROSS
CEO

MATT SCHMIDT
PRESIDENT & CTO

JESSE DAVIS
EVP & COO

KELLET ATKINSON
DIRECTOR OF MARKETING

MATT O'BRIAN
SALES@DZONE.COM
DIRECTOR OF BUSINESS DEVELOPMENT

ALEX CRAFTS
DIRECTOR OF MAJOR ACCOUNTS

JIM HOWARD
SR ACCOUNT EXECUTIVE

JIM DYER
ACCOUNT EXECUTIVE

ANDREW BARKER
ACCOUNT EXECUTIVE

CHRIS BRUMFIELD
ACCOUNT MANAGER

ANA JONES
ACCOUNT MANAGER

PRODUCTION

CHRIS SMITH
DIRECTOR OF PRODUCTION

ANDRE POWELL
SENIOR PRODUCTION COORDINATOR

G. RYAN SPAIN
PRODUCTION PUBLICATIONS EDITOR

ASHLEY SLATE
DESIGN DIRECTOR

SPECIAL THANKS to our topic experts, Zone Leaders, trusted DZone Most Valuable Bloggers, and dedicated users for all their help and feedback in making this report a great success.

WANT YOUR SOLUTION TO BE FEATURED IN COMING GUIDES?
Please contact research@dzone.com for submission information.

LIKE TO CONTRIBUTE CONTENT TO COMING GUIDES?
Please contact research@dzone.com for consideration.

INTERESTED IN BECOMING A DZONE RESEARCH PARTNER?
Please contact sales@dzone.com for information.

Executive Summary

It's 2016, and application security is still a major concern within DZone's audience of developers and tech professionals. To help improve the understanding of application security best practices and threats, DZone surveyed over 500 IT professionals on their approaches to security, how security is handled at their companies, and their concerns on securing applications. We've also asked several subject matter experts to share their thoughts on application security, particularly on the theory of automated security systems, responsibility, and the Internet of Things, which has become particularly relevant in the wake of Dyn's east coast DDoS meltdown.

ENTERPRISE SECURITY HAS A LONG WAY TO GO

DATA 15.3% of respondents say that they have no formal security testing, and 48% feel that testing is never sufficiently covered. 26.8% report no security training while 33.9% have had ad-hoc, informal training.

IMPLICATIONS Security issues continue to rattle established enterprises and make news, as was the case with Dyn's DDoS attack in 2016 and Target's holiday card hacking in 2014. However, formal security practices in the enterprise are still lacking. The causes for this lapse are unknown, whether it's related to cost, time, or expertise. However, it seems that about a third of respondents are training each other on security practices in lieu of formal training. In addition, only a slight majority of respondents feel that security testing is sufficiently covered.

RECOMMENDATIONS Regardless of any expense, enterprise development teams need to invest more into security, whether using established methods, such as the OWASP Top 10, or creating their own custom materials. While developers may be training each other, unified, company-led efforts are important to ensure that everyone is communicating well and on the same page. While current efforts are still lacking, the importance of security is being recognized. When asked to rank considerations between performance, security, maintainability, and scalability, those surveyed selected security as their second highest priority behind application performance.

WEB APPS CAUSE THE GREATEST CONCERN

DATA When asked to grade languages on security concerns, JavaScript was the most worrying language with an average score of 6.9 out of 10, followed by PHP with a score of 5.8 (median scores of 8 and 7, respectively).

IMPLICATIONS As JavaScript is notorious for the massive number of libraries and frameworks available to it, developers are most concerned about securing JavaScript apps. This contrasts with how respondents felt about languages like Objective C and Swift (3.5 each), both of which are typically used by themselves to create native mobile apps.

RECOMMENDATIONS The first step is to learn how to defend against two common JavaScript vulnerabilities: cross-site scripting (XSS) and cross-site request forgery (CSRF). You can find our fun explanation of these attacks and basic defenses in our infographic on page 20. Two of our articles also discuss app security that is relevant to JavaScript. In Wilfred Nilsen's article on page 6, he goes into detail about IoT (another technology known for its massive number of moving parts) threats and defenses, and stresses that if one defense measure falls, another should automatically take its place. Mike Milner makes a similar point on page 14: automated security measures are the most effective option.

DEVELOPERS ARE GETTING INVOLVED

DATA 50% of respondents believe that developers should be primarily responsible for security, while 28% put the task in the hands of security teams, and 22% believe frameworks should be responsible.

IMPLICATIONS The number of those who think developers should handle security are around the same as they were last year at 53%. While all three are responsible in some part for implementing security measures, there's still some education to be done about the importance of security throughout the software development lifecycle.

RECOMMENDATIONS As many of our data points have indicated, security measurements such as training and testing have been lackluster considering the potential impact of an attack. Whether this is due to cost, expertise, or a lack of concern is to be determined, but there should be communication amongst departments about the risks and a strategy to deal with them, regardless of who is responsible. Respondents are on the right track in recognizing that security should be built into a product, though the lack of significant movement is slightly concerning. In "App Security Is a Stack" on page 18, Lori MacVittie bridges the communication gap by reviewing the application, protocol, and platform layer of applications and whether the operations, network, or development teams should be responsible for handling them. A slight spoiler: every team is responsible for each layer.

Key Research Findings

517 software professionals completed DZone's 2016 Security survey. Respondent demographics include:

- 74% of respondents identify as developers/ engineers (42%) or development team leads (32%).
- 72% of respondents have 10 years of experience or more as IT professionals. 46% of respondents have 15 years or more.
- 41% of respondents work at companies headquartered in Europe; 30% work in companies headquartered in the US.
- 22% of respondents work at companies with more than 10,000 employees; 27% work at companies between 500 and 10,000 employees.
- 60% identify as developers or developer team leads.

WITH GREAT POWER...

50% of survey respondents said that primary responsibility for application security should be in the hands of developers, over security teams (29%) and frameworks. This is in keeping with the results of our 2015 security survey, in which 53% of respondents put security on developers first. Furthermore, respondents in development roles (74%) skewed slightly above the overall average of respondents who believe that developers are primarily responsible for AppSec, showing a broad understanding of the importance of security throughout the development process and a resistance to finger-pointing and scapegoating in the SDLC.

While having a primary team responsible is important for transparency and accountability, it is also important to know that application security is a concern for everyone, and the responsibility for creating secure applications should be accepted by all parties involved from design to dev to ops.

APPSEC BAG OF TRICKS

While there is no single right answer for how you should develop a secure application, there are some definite trends among our respondents regarding the practices, techniques, and tools they use. For application architectural patterns, the use of roles and sessions were both very popular among our respondents, with 77% and 75% claiming to use these patterns, respectively. For verifying message integrity, 68% of respondents said they use authentication tokens (including digital signatures). 70% of respondents said they use the OpenSSL toolkit for implementing encryption in their applications. And in terms of secure coding 64% of respondents said they consider security while architecting and designing an application, while 90% said they use input validation to help secure their application.

SECURITY TESTING HAS BEEN WEIGHED, HAS BEEN MEASURED, AND HAS BEEN FOUND WANTING

What is the main type of application security testing you do? Penetration testing and security code review were popular responses among our survey respondents this year, with 24% and 19% of responses, respectively. However, about one in six respondents said they have no formal security testing at all. Even when applications are tested for security, that testing may be inadequate. When asked how sufficient testing is determined for their applications, 15% of respondents answered "the attack surface is addressed to some acceptable level;" 15% said "the threat model is handled in an acceptable way;" and 22% responded that both of these occur. But 48% of respondents answered that "security testing is never sufficiently covered."

SECURITY TRAINING COULD USE SOME WORK, TOO

When asked how frequently developers in their organization are trained in security, 27% of survey respondents said that no such training occurs. 33% of respondents said

WHAT IS THE MAIN TYPE OF APPLICATION SECURITY TESTING YOU DO?

%	
24	Penetration Testing
19	Security Code Review
13	Vulnerability Assessment
16	No Formal Security Testing
4	I Don't Know
2	White Box Testing
6	Black Box Testing
15	Source Code Analysis
2	Dynamic Analysis

OF THE APPLICATIONS YOU TEST FOR SECURITY, HOW IS SUFFICIENT TESTING DETERMINED?

%	
15	The attack surface is addressed to some acceptable level
14	The threat model is handled in an acceptable way
48	Security testing is never sufficiently covered
22	Both 1 and 2

that security training occured in an ad-hoc manner in their organization, leaving 39% of respondents whose organizations have organized security training on a yearly, or more frequent, basis. Of the training discussed, organizations lean towards completely custom training; 66% of respondents said that their organization uses custom training materials, while 46% say that the OWASP Top 10 is used, and only 16% say that the SANS Top 25 is used for training purposes within their organization.

SECURITY STILL TAKES A BACK SEAT TO PERFORMANCE

Last year, between performance, maintainability, scalability, and security, respondents on average ranked security as third in terms of importance, behind performance and maintainability as numbers one and two. This year, security passed maintainability, but still lags behind performance in respondents' priorities. The gap, however, is closing as the importance of application security is increasingly acknowledged, and the gap in rank distribution between performance and security responses has shrunk considerably from last year.

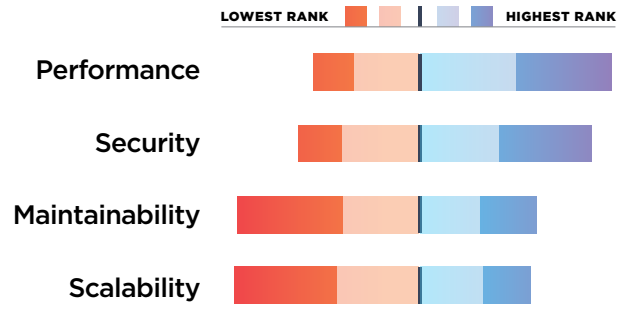
TO RELEASE OR NOT TO RELEASE...

On average, respondents to our 2016 Application Security survey said that 20% of application releases contained known security vulnerabilities, with a median response of 10%. This aligns closely with last year's responses, which estimated an average of 22% of releases containing known security vulnerabilities with a median response of 13%. 78% of respondents said that application security could override security concerns with half of respondents saying that this occured either sometimes (30%), often (14%), or all the time (5%), rather than rarely or never. 10% of respondents did not know how often application security was overridden by release schedules. Also, 50% of respondents said that security patches had to be released once every two months or more frequently, while 30% of respondents said that security patch release was not applicable to their application/situation.

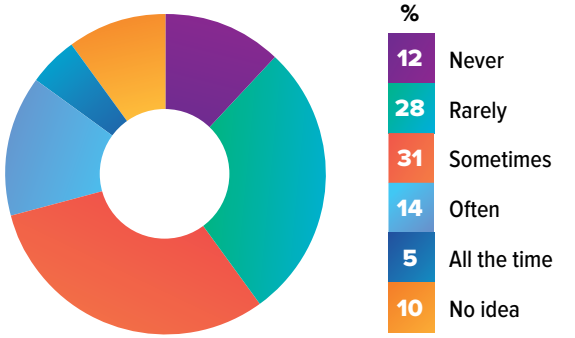
WEB APPS ARE CONCERNING

We asked respondents to rate how worried they were about security among several popular programming languages, on a scale from 1 to 10 (10 being maximally worrying). Results showed that the most concerning languages were JavaScript, with an average concern rating of 6.9 out of 478 responses, and PHP, with an average concern of 5.8 out of 341 responses (median responses were 8 and 7, respectively). Third most concerning was Java, with an average concern rating of 4.6 out of 504 responses. The inherent interconnectivity of web apps makes some sense of these concerns, but interestingly, languages used primarily for mobile applications, such as Objective-C and Swift, had very low ratings of concern (each had an average rating of 3.5 out of 298 and 293 responses, respectively, with median responses of 3).

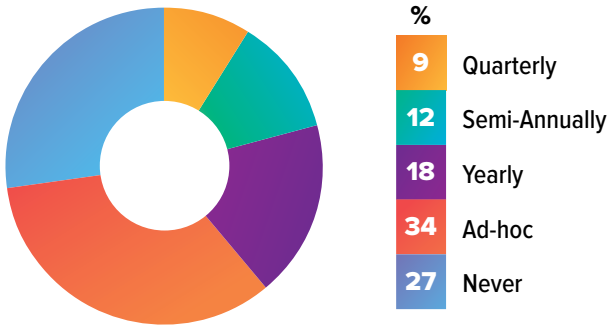
HOW DO YOU PRIORITIZE THE FOLLOWING CONSIDERATIONS? RANK FROM MOST IMPORTANT TO LEAST IMPORTANT.



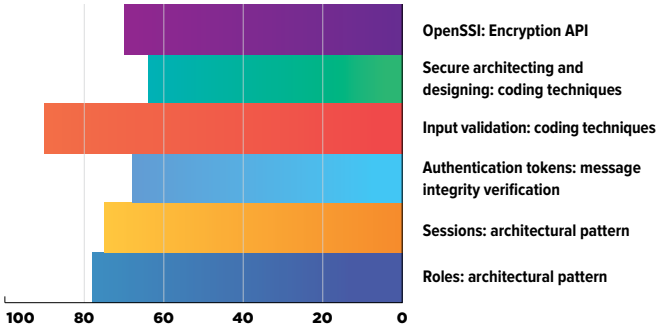
HOW OFTEN DO RELEASE SCHEDULES OVERRIDE SECURITY CONCERNS?



HOW FREQUENTLY ARE DEVELOPERS AT YOUR ORGANIZATION TRAINED IN SECURITY?



COMMON SECURITY PATTERNS, TECHNIQUES, AND TOOLS



Have We Forgotten the Ancient Lessons About Building Defense Systems?

BY **WILFRED NILSEN**
CTO AT **REAL TIME LOGIC**

QUICK VIEW

01 Start thinking about security from the earliest design steps. It must be integral to the complete solution.

02 Avoid designing a system that hinges on only one line of defense since the ramifications can be catastrophic if the one and only defense should go down.

03 Make sure you budget for implementing security. Expect to use third party products designed specifically for improving security.

I find the multilayered defense system built into many ancient forts fascinating. Fort Dún Aonghasa in Ireland is a great example of how a multilayered defense system was used to increase security by making it very difficult for an attacker to gain access to the "inner circle." The attacker would have to overcome several barriers, including first having to climb up a cliff, then penetrate huge boulders while being shot at with arrows, and then climb over three walls.

Just as many of the old forts used a multilayered defense system, so can a modern IoT solution. However, the reality is that many modern IoT solutions completely lack any type of defense system against hacking.

The lack of defense systems or the limited set of defenses found in many modern IoT devices and solutions often stems from engineers and designers lacking awareness of the vulnerabilities that may exist in certain IoT protocols. We will explore some of the typical weaknesses found in IoT devices/solutions and how IoT devices and infrastructure can be designed to be more resilient by deploying a multilayered defense system.

HACKED AND ENSLAVED IoT (SERVER) DEVICES OPERATING IN A BOTNET

Recently, we received the following email from our hosting provider:

Event Summary: A software anomaly was corrected that caused excessive outbound routing announcements to be withdrawn in response to a Denial of Service attack.

Distributed Denial of Service (DDoS) attacks are on the rise, and the origin of these attacks are increasingly coming from hacked IoT devices. Just recently, Dyn, a company that provides DNS services for companies like Twitter, Amazon, Spotify, and Netflix was hit with a record DDoS attack, causing outages and network congestion for some of the internet's most popular websites. The attack originated from a botnet of thousands of hacked and enslaved IoT devices such as IP cameras and DVRs from all over the world.

All of these hacked IoT devices have one thing in common—they all operate as servers by providing services such as web-server, telnet server, and/or SSH server. Unfortunately, publicly accessible servers can easily be found by port scanners such as Shodan. A hacker can create automated tools that scan and probe for weaknesses such as easy-to-guess passwords, default product passwords, or simply performing brute force password attacks. Hackers then upload code to the compromised devices, thus enslaving them by integrating these devices into their botnet. For this reason, IoT devices that include services are much more vulnerable than IoT devices that provide no form of publicly available service.

FIRST LINE OF DEFENSE: IoT DEVICES SHOULD OPERATE AS CLIENTS, NOT SERVERS

Devices operating as network clients—as opposed to

operating as network servers—cannot be found by port scanners. In addition, when devices operate as clients, it's impossible for an outsider to directly connect to the devices. However, devices that operate as clients need an online server that enables the users to control their devices via the online service.

When all devices and human machine interfaces operate as clients, an online server is required for proxying the traffic between the users and the devices. The online server operates as a service/server to the human machine interfaces (HMI) and to the connected devices. For example, a homeowner with a cloud-enabled thermostat can control the thermostat via a web interface provided by the online server.

For example, although a proxy server can be designed by using the WebSocket protocol and by designing server-side code for routing (proxying) the messages between users and their devices, an easier solution may be to use one of the many IoT protocols designed for this purpose. Publish/subscribe (pub/sub) protocols, such as AMQP, XMPP, and MQTT, are popular choices.

An online proxy server is typically referred to as a broker when using pub/sub protocols. The broker is in charge of routing messages between publishers and subscribers.

IoT PUB/SUB PROTOCOLS AND THE HIDDEN PINHOLES

Pub/sub protocols are great choices for controlling devices indirectly via an online server. They make it possible for any connected client to subscribe to topics. Some IoT protocols, such as MQTT, also enable what is known as wildcard subscriptions. A wildcard subscription lets a client subscribe to topics without knowing the exact topic name, thus a potential pinhole. It is in fact possible to subscribe to any topic in MQTT making the protocol inherently insecure since an attacker that has gained access to a broker can eavesdrop on all messages sent from other devices. The attacker can then learn the details of all messages used by the IoT solution and use this information to indirectly compromise all connected devices by publishing specially crafted messages.

A DEFCON MQTT paper was recently released by a white hat hacker. The paper reveals how one can find and access MQTT brokers on the Internet and perform actions such as open prison doors, change radiation levels, and so on. The online brokers that the hacker refers to are not requiring the MQTT clients to authenticate. The hacker then goes on to create a script that subscribes to all messages handled by the brokers by using wildcard subscriptions.

Needless to say, a protocol such as MQTT cannot be used without client authentication. However, since MQTT sends passwords from the client to the broker in clear text, the communication must also be protected by TLS to protect from eavesdropping. Note that using client authentication may not be as secure as you think. We will explore the authentication security issues in the next two sections.

> The lack of defense systems or the limited set of defenses found in many modern IoT devices and solutions often stems from engineers and designers lacking awareness of the vulnerabilities that may exist in certain IoT protocols

SECOND LINE OF DEFENSE: AUTHENTICATION

IoT devices operating as clients should use the TLS protocol for authenticating the server at connection time. When the client connects, the online server's X.509 Certificate provides assurance that the device is in fact connecting to the correct server and not to a spoofed system.

As mentioned previously, pub/sub protocols such as MQTT should not be used without client authentication since this would compromise the IoT solution. Client side authentication in combination with server side authentication is known as mutual authentication. Server authentication is normally provided by the TLS protocol and the server's certificate. However, a client can authenticate itself by using anything from a plain text password to an X.509 Certificate.

All forms of authentication mechanisms are based on keeping a secret. Keeping secrets in IoT devices such as headless edge nodes (a device that lacks a graphical user interface) is problematic, for these devices are usually out in the wild, thus enabling a hacker to potentially extract the hardcoded credentials from the device. For example, a cloud-enabled thermostat that is designed to connect to an online cloud server can be purchased by a hacker, who may then extract the credentials (password or X.509 Certificate) from the device, thereby gaining access to the thermostat's online ecosystem. This is particularly concerning for IoT solutions based on pub/sub protocols that enable wildcard subscriptions.

AUTHENTICATION SHORTCOMINGS

When using a pub/sub protocol such as MQTT, a hacker that manages to extract the credentials (password or X.509 certificate/private key pair) from a device can

use the credentials for either eavesdropping on the IoT solution's communication or performing a direct exploit by publishing specially crafted messages.

Unique credentials per device makes it possible to disable the particular exploited device including use of the device's credentials; however, this requires that the IoT solution can detect the exploit and remove the exploited credentials from the solution. A solution that uses an X.509 certificate/private key pair for client authentication is even more complicated since compromised X.509 certificates must be managed by using a certificate revocation list (CRL).

The complexity in extracting the credentials from a device greatly depends on the device type and the components used in the device. A device based on a high-level operating system where the credentials are stored on a file system in an external flash memory module makes it much easier for a hacker to extract than the credentials for a device using internal microcontroller flash memory and where the JTAG fuse is blown. Having said that, even the most hardened device can be exploited and the credentials extracted. For this reason, designing an IoT solution that relies on using credentials as the only defense mechanism is going to be much more vulnerable than an IoT solution based on a multi-layered defense protection system.

The ancients taught us that a multilayered defense system improves overall security. If one defense fails, another takes over.

THIRD LINE OF DEFENSE: AUTHORIZATION

Authorization protects the IoT solution against compromised devices when the credentials have been extracted and used by a hacker and for IoT solutions designed to be used without password protection. Authorization can also be used to detect abnormalities in the communication pattern and report such incidents to an operator.

Authorization is particularly important for protocols, such as pub/sub, that provide the one-to-many message model. Authorization is even more important for pub/sub protocols that enable wildcard subscriptions.

Authorization is product specific and can come in many flavors such as providing a method for controlling an Access Control List (ACL). IoT solutions based on pub/sub protocols that enable programmatic authorization on the server side, for example by providing a plugin system where you can use your own computer code for analyzing the traffic, can be made more secure than a broker solution that only enables authorization via configuration files.

SUMMARY

The ancients taught us that a multilayered defense system improves overall security. If one defense fails, another takes over. We should take this as a history lesson and apply it to modern day IoT and network design.

First, devices should operate in stealth mode, making them invisible for automated hacker bots searching for devices. A device operating as a network client has stealth mode property. As an added security feature, an IoT solution that also operates the online server in stealth mode is more secure because an attacker would have a hard time finding the online service. The WebSocket protocol has this property for it is difficult to differentiate a WebSocket server from a regular web server, especially if the entry URL for the WebSocket server is nonpublic since a nonpublic URL cannot be found by an automated port scanner. Protocols that only provide one type of service and that listen on a specific port number do not have stealth mode property.

Second, IoT servers should use X.509 certificate authentication to prevent man in the middle attacks. In addition, some IoT protocols should not be used without client-side authentication since they include features such as wildcard subscriptions that may jeopardize the security of the entire solution.

As a third line of defense, the server/broker should include authorization to protect the IoT solution. Servers that enable custom and programmatic analysis of the messages include additional security that makes it possible to provide fine-grained authorization and detection of non-conforming messages.

Whatever protocol you choose, a good understanding of the protocol will help you design a better defense system for your IoT ecosystem. Many protocols include a wealth of features, however, these features, used or not, may lead to pinholes that can be used by attackers attempting to compromise your solution. A recommendation is to choose a protocol with the right set of features designed with security in mind rather than a protocol supporting everything that is more attack-prone.

WILFRED NILSEN is a computer programmer focusing on securing IoT. When not working or with the kids, you'll find him paddling in and outside of Dana Point harbor.

Agile and Secure?

Development and Security CAN Work in Peace and Harmony

There has been a lot written about DevOps and Continuous Delivery practices and how to apply these practices to software development to improve efficiencies. With the growing momentum around adding secure coding practices into the mix, White Hat Security has some recommendations on how static (SAST) and dynamic (DAST) application security testing solutions can help teams achieve their objectives, while also saving time and resources and reducing the security risk of deployed apps.

1. **Reduce the time required from definition complete to code release.** Model application development methodologies such as Agile, Scrum, or Kanban focus on releasing software early and often. The old days of PRDs and Waterfall approaches have been replaced with user stories as inputs to development. The best user stories clearly address any assumptions and constraints, as well as acceptance criteria. With this in mind:
    A. Define application security requirements (e.g., compliance to OWASP Top Ten, SANS 25, PCI-DSS, etc., or "no vulnerabilities rated medium or high risk") up front.
    B. Implement static application security testing (SAST) early in the development process and continuously fix security vulnerabilities as they are identified to help ensure that security requirements are met prior to code release.

2. **Improve the quality and speed of feedback for developers.** Developers need to get feedback on which software flaws and security vulnerabilities are present in their code, ideally as they are developing it, not months later when they've forgotten exactly what they did.
    A. Make sure that your AppSec testing solutions have plugins for bug trackers (e.g., Jira), so that vulnerabilities in scanned code are automatically synchronized and appear as tickets in Jira.
    B. Check whether plugins for continuous integration build tools, such as Jenkins, are available, so that automated security scanning of websites (DAST) and code (SAST) can be scheduled as often as needed as part of an Agile workflow.
    C. Use both SAST and DAST AppSec solutions to ensure maximum test coverage. Use DAST for continuous scanning at pre-production and production phases. Use SAST for development and QA phases. All identified security vulnerabilities should be input into Jira and tracked.

3. **Reduce rework.** "Fixing security bugs at design time costs 1/60th of what it costs to fix the same bugs with a patch after the release."[1]
    A. Use a SAST solution to not only save time, money, and resources, but also to reduce the security risk of your deployed apps. WhiteHat Security has seen instances where customers have experienced a 60% decrease in the number of Cross-Site Scripting (XSS) vulnerabilities detected in their deployed production apps, if they use a SAST solution during app development.

[1] **Source:** Secure Coding: Principles & Practices, O'Reilly Media, 2003

Automating Application Security in Modern Software Development

BY **JEFF WILLIAMS**
CO-FOUNDER AND CTO AT **CONTRAST SECURITY**

QUICK VIEW

01 Application security is the leading cause of breaches.

02 Fast, accurate, and scalable application security tools are the key to automating security.

03 Balance tools to assess vulnerabilities and tools that protect against attacks.

04 Look for tools that integrate directly into your development and operations environments.

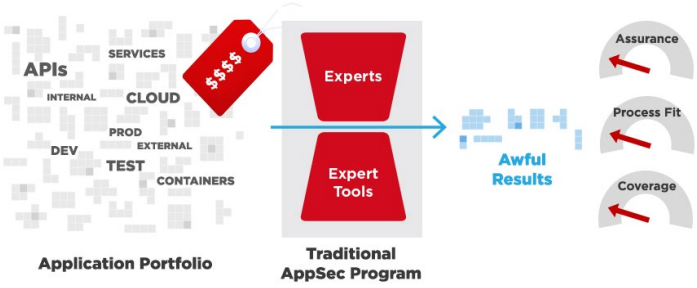
Today, every organization has become a software company. The increasing dependence on automation demands that software survive and thrive despite an increasingly hostile environment. Insecure code has become the leading security risk and, increasingly, the leading business risk as well. It's irresponsible at every level to ignore this risk while doubling-down on anti-virus solutions and firewalls — neither of which protects applications.

APPLICATION SECURITY DEMANDS AUTOMATION

As software increasingly "eats the world," the security of that software becomes increasingly important. Every line of code you write makes you easier to attack. Insecure software caused 82% of financial breaches in 2015 and has been the leading cause of breaches overall for the past nine years. Virtually every study has shown that almost every web application and API has serious vulnerabilities.

But the scale of the problem is out of control. We need tools that enable novice developers to reliably build and operate secure applications and APIs. We must transform paper-based security policy and guidance into "security as code" without disrupting modern, high-speed software development.

Unfortunately, even well-established application security programs often can't operate at the speed and scale required. These programs rely on experts and their tools are for use by experts only. The traditional approach disrupts the software lifecycle and is incompatible with modern high-speed software development. When you combine the technology and human cost, the annual per-application cost for these programs can range from $50,000 to $100,000 per year.



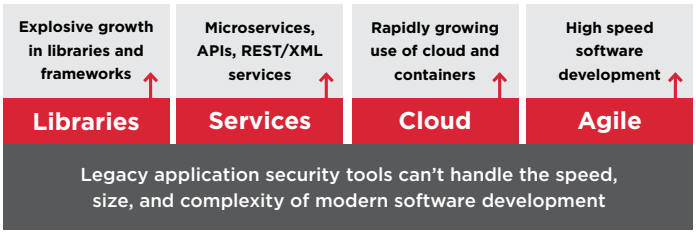
AND SOFTWARE IS GETTING INCREASINGLY DIFFICULT TO SECURE

Let's look at some of the factors that make securing modern software difficult.

First, there has been an explosion of frameworks and libraries since the inception of automated dependency resolution tools like Maven. This has resulted in applications with several hundred libraries instead of just a few. This massive increase in size makes it much more difficult to find vulnerabilities. So tools must be

aware of these libraries and how they are used by custom application code.

Further, there is also a trend towards using APIs (REST, SOAP, RPC, etc.) to create applications with Angular or mobile front ends. These APIs are difficult to assess and protect because the structure of their communications is more difficult to understand. Unless tools can understand the application, they don't produce very good results. And if the tools require a lot of expert tailoring and configuration, they won't get used.



Finally, high-speed software development practices like Agile and DevOps have broken traditional approaches to security. Organizations used to wait until just before deployment to do an in-depth security review that could take weeks to complete. However, when projects are deploying weekly, daily, or hourly, there's just no time for that approach. We need a different approach and different tools.

BEST PRACTICES FOR AUTOMATING APPLICATION SECURITY IN MODERN SOFTWARE PROJECTS

Modern software development requires *continuous* security to go along with continuous integration/delivery/deployment. The challenge in a nutshell is enabling an existing development pipeline to reliably produce secure software without creating roadblocks or even speedbumps. The reality is that if security slows down innovation, it will be bypassed.

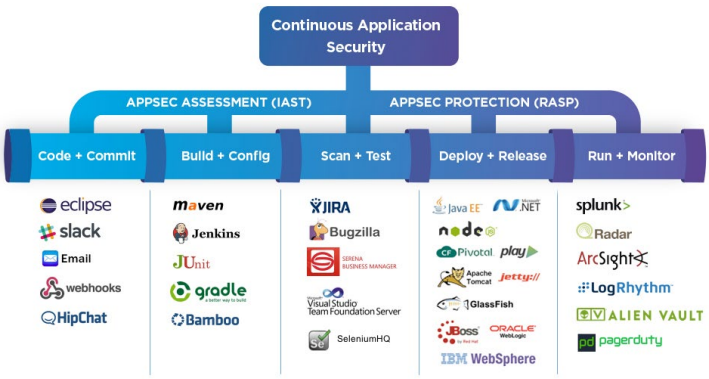
1. **Choose applications security tools for speed, ease-of-use, accuracy, and scalability.**
   Instant feedback and ease of use are critical. Appsec tools need to be usable by people in development and operations without any security experience. Any inaccuracy will require an expert to resolve, and experts don't scale.

2. **Integrate security directly into your pipeline.**
   To shorten those feedback loops, look for tools that deliver results directly into tools you're already using, like Slack, HipChat, JIRA, Maven, Jenkins, SIEM, and PagerDuty. Security issues should look and feel like any other kind of development or operations issues.

3. **Detect vulnerabilities.**
   Modern software development demands high-speed feedback on vulnerabilities. Legacy static (SAST) and dynamic (DAST) scanners are difficult to automate and generate false alarms. Investigate the use of newer interactive (IAST) tools that assess your applications from within, using the latest instrumentation technology.

4. **Protect against attacks.**
   Application attack protection isn't just for defense against known attacks, it provides a fast and flexible way to block novel attacks that emerge. Legacy web application firewalls (WAF) create network architecture complexity and aren't very accurate. Fortunately, runtime application self-protection (RASP) is gaining wide adoption for its flexible deployment and impressive accuracy.

5. **Use threat intelligence and security research to improve your security architecture.**
   Using generic tools that search for "negative" coding patterns is a good start. But as you mature, you may want your tools to automatically enforce the security patterns you've chosen. This is a "positive" approach to security. Ultimately, you want to be able to automatically verify that all your applications have the right security defenses in place, that all the defenses are correct, and that they have been used in all the right places.



The good news is that it is possible to create a software pipeline that can enable you to reliably secure code and protect applications in operations. Modern application security tools can give you instant feedback on both vulnerabilities and attacks.

JEFF WILLIAMS brings more than 25 years of security leadership experience as co-founder and Chief Technology Officer of Contrast Security, which is revolutionizing application security with self-protecting software. Previously, Jeff was co-founder and CEO of Aspect Security and a founder and major contributor to OWASP, where he served as Global Chair for 8 years and created the OWASP Top 10. Jeff has a BA from Virginia, an MA from George Mason, and a JD from Georgetown.

FAST TRACK SECURITY PROTECTION

Security fixes in minutes not weeks

bmc digital IT

bmc.com

Modern Solutions

In the wake of attacks on IT systems, the news invariably points to an opening that a hacker was able to exploit. And, it is frequently revealed that the hacker lurked in the system for months or years before the actual attack was identified and made public. These scenarios are the stuff of dreams for nefarious hackers and yet enterprises have a "why bother" attitude of coordinated IT security and believe that an attack is simply a matter of time. Unfortunately, in most enterprises today, the classic finger-pointing of IT security and IT operations is a reality. It is safe to say that IT operations does not understand the intricacies of what IT security does, and, IT security does not understand the intricacies of what IT operations does. This is not a fault of one team or the other. It is simply two distinct professional teams with different roles for achieving the same goal. IT security and IT operations must work more closely together to maintain secure operations. Enterprise organizations must set a mandate to operationalize security from the C-level to mitigate the business risk of security threats and vulnerabilities. Adopting solutions such as BMC's BladeLogic Server Automation, BladeLogic Network Automation, and BladeLogic Threat Director will help solve technical problems and form a well-coordinated offensive approach to solving ever-present security threats and vulnerabilities. Having solutions that are purpose-built for operationalizing security allows more visibility into the role of IT security and IT operations. Hackers are aggressive, intrusive, and invasive unwanted guests. Enterprises must combat the criminal vigorously, actively, and boldly to protect the business. Solutions such as BMC SecOps will help organizations defeat 21st century enemies.

> Too frequently IT security becomes a top priority only when an enterprise experiences a damaging security breach.

WRITTEN BY ALLISON CRAMER
DIRECTOR OF SOLUTIONS MARKETING, **BMC**

PARTNER SPOTLIGHT

BladeLogic By BMC

Eliminate risks and blindspots to reduce the attack surface with an action-focused solution

CATEGORY
Vulnerability and Patch Management, Compliance

NEW RELEASES
Continuous

OPEN SOURCE
No

CASE STUDY The Michigan Department of Technology, Management, and Budget (DTMB), an agency of the state government of Michigan, is well along in its digital transformation of IT—particularly in the area of citizen services. DTMB delivers centralized IT services to 18 state agencies and supports 300 online services for the state's 10 million citizens. To further this path to digital success, DTMB needed a next-generation infrastructure to enable faster response to service requests, enhanced operational and regulatory compliance, strengthened security, and streamlined audits across its diverse IT infrastructure.

As a central component of its digital transformation strategy, DTMB implemented BMC BladeLogic Server Automation to automate the management, control, and enforcement of server configuration changes in the data center. With BladeLogic, DTMB is accelerating server provisioning to speed fulfillment and ensuring operational consistency to create a more stable and secure environment. Detailed reporting identifies servers that need security patches, enabling more robust security and continuous compliance with regulatory and industry requirements.

STRENGTHS

- **Patching:** Supports and follows maintenance window guidelines to ensure timely delivery of patches
- **Remediation:** Links vulnerabilities to identified patches and creates a remediation plan though BladeLogic Threat Director
- **Compliance:** Integrates role-based access control, pre-configured policies for CIS, DISA, HIPAA, PCI, SOX, NIST, and SCAP, documentation, and remediation
- **Blindspot Awareness:** Identifies the areas of your infrastructure which are not being monitored, leaving you exposed and make adjustments leveraging integration with BMC Discovery
- **Powerful Dashboards:** Determines threat stature through the use of two BladeLogic Threat Director dashboards, designed specifically to suit the requirements of both the operations user and the security user. Finds performance trends, SLA compliance, and threat history, and quickly prioritize remediation activities.

NOTABLE CUSTOMERS

- SAP
- Fujitsu
- Morningstar
- State of Michigan
- Transamerica
- Lockheed Martin

BLOG bit.ly/2fy5f6O | **TWITTER** @BMCSoftware | **WEBSITE** bmc.com/secops

Frameworks Make Coding Easy and App Security Hard

QUICK VIEW

01 Development frameworks provide some basic security, but the developer must understand how it works for security to be effective.

02 Developers are focused on building product and features. They need tools to help identify security mistakes and to protect the application once it is deployed.

03 App security is a very dynamic field. Choose tools that emphasize automatic remediation and updates against new classes of vulnerabilities.

BY **MIKE MILNER**
CO-FOUNDER AND CHIEF TECHNOLOGY OFFICER AT **IMMUNIO**

It is easier than ever to write web applications, but educating developers on security issues hasn't kept pace with the evolution of the threat environment. Therefore, relying on developers to be infallible when it comes to web app security is an expensive, and losing, proposition.

The truth is that creating secure software should not require that developers become specialists in security. The most effective way to secure web applications is to make security a fundamental part of the software—to modify the application framework so that apps automatically defend themselves against common vulnerabilities. There are challenges inherent in this process, but this simple change in approach can fundamentally improve application security.

PROBLEM 1: FRAMEWORKS MAKE CODING EASY AND SECURITY HARD

EXAMPLE 1: RAILS HELPERS AND SAFE_BUFFERS

The framework does some basic work to keep you secure. Unfortunately, it also makes it very easy to unintentionally bypass your own security. Safe buffers are designed to protect against cross site scripting. Unfortunately, in older versions of Rails, it is possible to add an unsafe string to a safe buffer and, therefore, send untrusted content that appears to the system to be safe.

EXAMPLE 2: RAILS DIRECTORY TRAVERSAL

In the last decade, directory traversal (also called path traversal) has been one of the top five vulnerabilities in applications built on the Ruby on Rails framework. This vulnerability enables users to gain access to files located outside the directories within an application to which a user legitimately has access—potentially including critical system files or source code.

The main way to prevent against directory traversal and abuse of safe buffers is educating developers about situations in which framework security isn't enough, provide automated tools to add defenses, and implementing defense in depth.

APPLICATION DEFENSE IN DEPTH

As the above examples indicate, while it is easy to build a web application, securing the application, and the network it runs on, is not. Hackers know that web apps can be the best way into an organization and they take advantage of the vulnerabilities that apps introduce into a system. According to Verizon [1], 40% of all confirmed breaches in 2015 were the result of attacks on web applications.

Application defense in depth means using layers of defense within the application and the application layer protocols. If an attack makes it through a layer—from inside or outside your inner trust boundaries—it cannot compromise the system.

THE TROUBLE WITH WAFS

Web application firewalls (WAF) do a reasonable job protecting traditional web applications and fighting against cross-site scripting (XSS) and SQL injection (SQLi). The problem is that WAFs can be complicated to setup and easy to bypass (because they only protect network traffic routed through them). They protect from outside an application, so they have no visibility into what is happening inside the application and cannot protect against a threat coming from within. They also can protect only against known vulnerabilities and attack signatures. Once a threat evolves or a hacker finds a workaround, the application is again vulnerable.

SECURITY IS A FRAMEWORK RESPONSIBILITY

The threat environment evolves rapidly, and software development cycles are shortening and becoming even more iterative, as companies release early versions to be tested in the marketplace. Because of that, it is difficult for developers and security teams to keep up to date with security needs. Thus, the most effective security comes from within an application framework, rather than a gatekeeping program or a security protocol that layers on top of an application or one that needs to be constantly updated to adapt to new threats.

PERFECT CODE IS A PIPE DREAM

No matter how accomplished the developer, it is almost impossible to keep up with the rapid changes in the threat environment and the security protocols required for effective defense. And the truth is that most developers don't learn much about security unless they have to. Even with a top-notch development team, sophisticated processes, strong quality assurance practice, and robust testing, applications are still released with significant (sometimes already identified) vulnerabilities.

BUILDING SELF-DEFENDING FRAMEWORKS

The most effective security comes from within. These apps are aware of the look and feel of normal operations and can identify unusual or malicious behavior and protect themselves when that happens. Because of this, they do not need constant updating to address new or evolving threats.

PROBLEM 2: APPLICATION DEPENDENCIES

Applications require third party components, leaving them vulnerable because of poor secure coding practices. This enables attackers to choose from many different attack vectors and disguise them from traditional network protection appliances.

PROBLEM 3: HTML IS A HORRIBLE MISHMASH

Unlike more traditional development environments that require consistency checking and compilation, modern script-based technologies like HTML5, CSS, and JavaScript are much more forgiving when it comes to writing executable code. If adoption of these technologies is not followed with a significant improvement in quality assurance, the result is often erratic and vulnerable code.

Web applications are a magnet for vulnerabilities like cross site scripting (XSS), which is one of the most prevalent vulnerabilities. It is easy to introduce into the template; and tedious to remediate, because each vulnerable field (potentially thousands in a single application) requires custom remediation code.

PROBLEM 4: GENERALIZING THE APPROACH

Over and over again, new zero day vulnerabilities are found in different parts of the vast technology stack. And most of the time these vulnerabilities are fueled by some critical and inherent weaknesses in the technology. SQL injection is a good example and has been one of the top application risks for many years, because of deeply rooted security weakness in database software.

PROBLEM 5: STRING BUILDING

Some prevalent code patterns, like search functionality or report generation, can be challenging to secure because they must combine a user friendly interaction with preventing unauthorized access to data.

TODAY, SECURITY IS AN AFTERTHOUGHT

The hard to hear truth is that the speed of development means that securing applications is often not top of mind when developing new web applications. For some organizations, it's because they don't have the resources or expertise. For others, it is due to a misdirected belief that web applications can be secured with other programs and protocols.

To be most effective, developers and organizations need to consider security at every step of the application lifecycle, including oversight and sufficient training, to enable security to matter from start to finish:

DESIGN

- designing security features
- using secure coding standards
- taking advantage of the security features in languages and application frameworks

TEST

- including static, dynamic and interactive analysis (SAST, DAST, IAST)
- penetration testing
- bug bounties

REPAIR

- fixing breaches
- remediating vulnerable code
- patching
- runtime application self-protection.

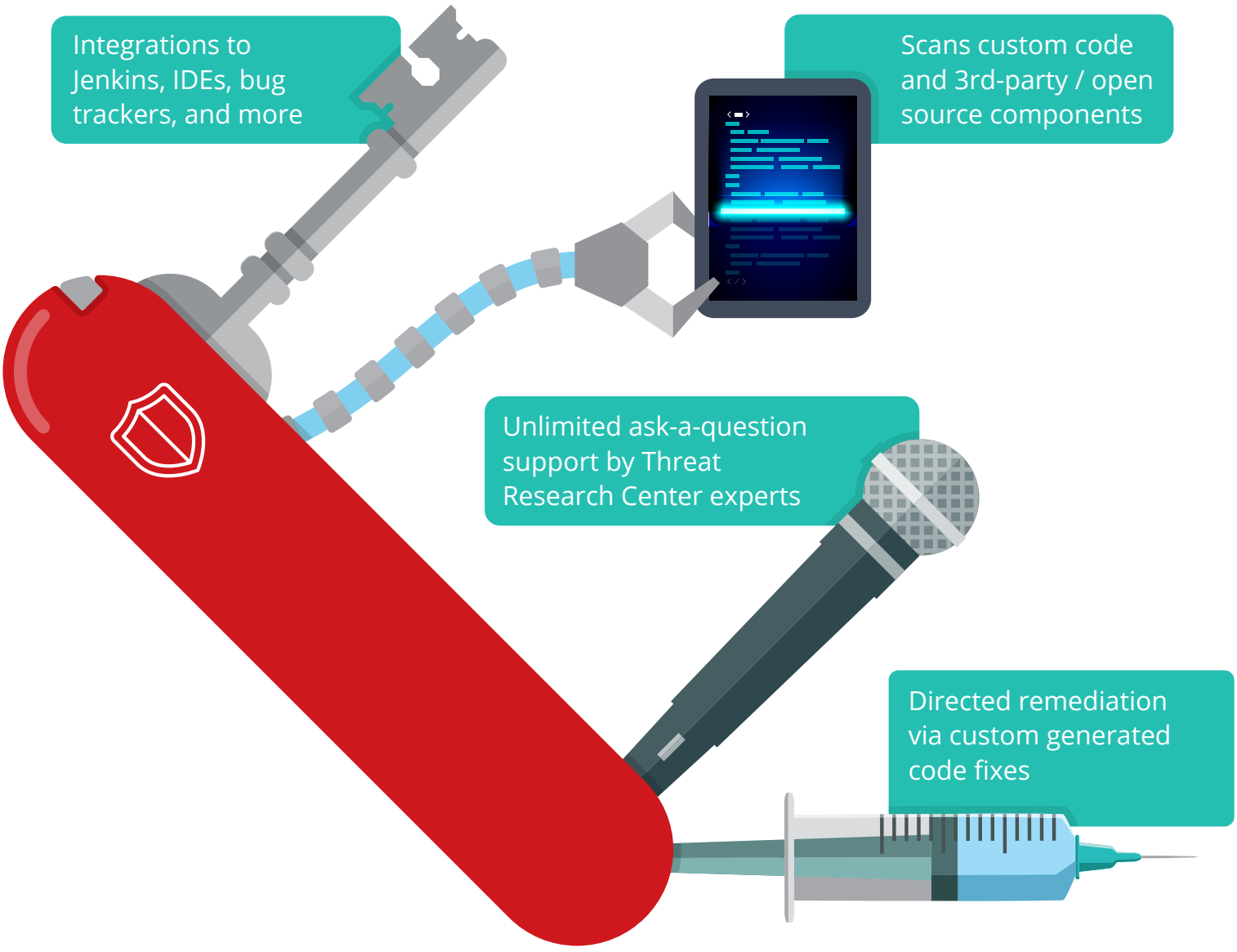
CONCLUSION

True security for web applications requires a new understanding of where the threats come from and how vulnerabilities affect their operation. To truly improve application security, organizations must focus on understanding and controlling what is inside their influence—the application and its execution. The most efficient and effective way to do that is to enable applications to automatically defend themselves.

[1] Verizon 2016 Data Breach Investigations Report, http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/]

MIKE MILNER is the Co-Founder and CTO at IMMUNIO. Between fighting cybercrime for the Canadian government and working for security agencies overseas, Mike has developed a deep understanding of the global security landscape and how the underground economy dictates hacks and ultimately drives breaches. Prior to founding IMMUNIO, Mike was a lead member of the technical staff at Salesforce.com and served as a software engineer at Canonical, following his time serving both the Canadian and UK Governments.

DevOps Security in Your Pocket



WhiteHat Security provides you with the resources to secure your web apps.

LEARN MORE

WhiteHat SECURITY

www.whitehatsec.com

Integrating Security into DevOps Workflows

How do we get developers, security engineers, and DevOps engineers all on the same page so we can implement successful application security programs? Considering that Web application attacks represent the greatest threat to an organization's security -- 40% of breaches in 20151 -- this is not a trivial question.

The problem is that these roles look at application security from different perspectives. Developers need to hit tight app development deadlines and are not always motivated to fix security vulnerabilities in their code, especially when they don't understand exactly how to do it. Security engineers who understand security vulnerabilities may lack the coding expertise to give guidance and help developers fix them. And DevOps engineers need security testing to be automated and support continuous integration workflows.

The right application security platform should facilitate a strong working relationship between security and DevOps towards the creation of more secure products. The solution should:

- Integrate with key developer tools, IDEs (e.g., Eclipse, Visual Studio, IntelliJ, and Xcode), source code repositories, bug trackers such as Jira and Bugzilla, and build servers like Jenkins and ALM tools (e.g., HP ALM and IBM Rational Team Concert). WhiteHat Sentinel Source, for instance, can be applied early in the SDLC for development and QA stages, and WhiteHat Sentinel Dynamic can be used for security testing in the pre-production and production phases.
- Provide accurate and actionable security reports (with near-zero false positives), along with custom security vulnerability descriptions and remediation advice to help developers understand and fix security vulnerabilities and acquire secure coding skills.
- Offer unlimited access to security experts, where developers and security engineers can ask a question about a specific vulnerability and receive a response, typically within 24 hours, without incurring extra support costs.

[1] *Verizon 2016 Data Breach Investigations Report*

WRITTEN BY ANNA CHIANG
SENIOR MANAGER, PRODUCT MARKETING, **WHITEHAT SECURITY**

PARTNER SPOTLIGHT

WhiteHat Sentinel Source By WhiteHat Security

WhiteHat Security combines technology and human intelligence to deliver the industry's most accurate application security solutions.

CATEGORY
Vulnerability and Patch Management, Compliance

NEW RELEASES
Continuous

OPEN SOURCE
No

OVERVIEW

WhiteHat's Sentinel Source static application security testing (SAST) solution scans your entire source code for vulnerabilities and provides detailed, prioritized results to help you determine where to best allocate resources based on severity and threat value.

Sentinel Source enables you to:

- Assess code at any point in the development cycle – even partial code.
- Run scheduled assessments daily or on demand.
- Preserve your intellectual property and scan source code on your premises (cloud option also).
- Achieve dramatic improvements in productivity for faster time-to-market.
- Reduce the occurrence of vulnerabilities found in deployed applications by fixing them first in development, when it's least expensive.
- Improve the security of your deployed applications and lower your application risk and exposure.

STRENGTHS

- Accurate and actionable vulnerability results due to expert review of scans by security engineers in WhiteHat's Threat Research Center (TRC); near-zero false positives saves time and effort
- Fast scanner can scan multiple apps concurrently; typical TRC response time is less than 24 hours
- Provides prioritized vulnerabilities and custom vulnerability descriptions and remediation advice
- Integrations with developer tools, IDEs, bug trackers and CI workflows improves productivity
- Software Composition Analysis (SCA)

SERVICE INCLUDES:

- Fast unlimited scans
- TRC-verified results
- Scanner tuning and rule pack updates
- Unlimited ask-a-question to the TRC
- Software Composition Analysis (SCA)
- Directed remediation with custom-generated code fixes

BLOG whitehatsec.com/blog | **TWITTER** @whitehatsec | **WEBSITE** whitehatsec.com

App Security is a Stack

BY **LORI MACVITTIE**
PRINCIPAL TECHNICAL EVANGELIST AT **F5 NETWORKS**

QUICK VIEW

01 Apps today are the backbone of the digital economy and require constant vigilance to defend against attacks.

02 Apps are threatened by a variety of attacks that target data, logic, protocols, and platforms.

03 App security is not just for developers anymore. Securing apps takes the cooperation of dev, ops, and network teams to identify potential threats and implement architecturally appropriate solutions.

App security has become as nebulous a term as "cloud" and "DevOps". There are a variety of application-related security concerns that must be addressed across development, operations, and the network that all fall under this single moniker. To better understand app security, it becomes important to recognize the different layers that comprise application security – the app security stack, if you will – by defining the parameters of each layer and assigning some measure of responsibility to the various teams who must take up its banner.

This is increasing important as attacks continue to expand their reach beyond what developers control — the application itself — to include what developers do not necessarily control - platforms and protocols.

Roughly that means the application security stack is built from three distinct layers: application, protocol, and platform.

APPLICATION LAYER

The application layer concerns itself mostly with application logic and data handling. Application layer security revolves around concepts such as validation of input and output and logic flow. The OWASP Top Ten is an excellent resource for understanding the most common application layer security challenges and includes the most commonly exploited vulnerabilities such as SQLi and XSS, among others. While these common security mistakes can be addressed with solutions upstream from the application – such as web application firewalls – they are best resolved in application code by careful application of secure coding practices.

Logic errors are more difficult to resolve outside the application. That's because the logic dictating flow between pages, tasks, and processes are generally fully contained within the application code itself. Ensuring proper flow and preventing users from abusing the ability to manually key in the URIs associated with most applications (and APIs) today is paramount to preventing access without the proper data available. Also falling into this category is prevention of parameter tampering. Applications relying on cookies or query parameters in URIs are particularly susceptible to this form of attack. While stateless application design is becoming preferable to traditional stateful design, careful attention to maintaining parameter integrity should be a key design factor, recognizing that client-side state is more vulnerable to this type of attack than its server-side alternatives. Encryption of client-side parameters can help prevent tampering, but there are operational consequences to adding this layer of security to the architecture.

Application client-side security is a growing issue, as the ability to inject malicious, obfuscated code into web applications via compromised ads and browsers has become almost trivial. This is an insidious, difficult to detect attack on applications as it often occurs on unmanaged client software over which neither developers nor operations has any control. There are a variety of mechanisms to combat this type of attack, both clientless and agent-based, which should be considered to prevent hijacking of this often-overlooked part of most applications.

Primary Responsibility: Developers
Secondary Responsibility: Operations / Network

PROTOCOL LAYER

The protocol layers of an application have become a critical concern for those dealing with application security. It is difficult to split protocols from the platforms that support them. In the

case of application security, we can do so by identifying the difference between exploitation of protocol behavior from exploitation of platform handling. Protocol behavior deals with the expected behavior of a given protocol. For example, HTTP is a reply/request based protocol in which a client sends a request and expects a response. Protocol exploitation involves using this proper behavior to cause an unexpected result.

HTTP-based Distributed Denial of Service (DDoS) rely on nothing more than proper behavior of HTTP. With flood-type attacks, a high volume of HTTP GET requests is sent to the application in question. These requests are legitimate, valid HTTP GET requests but are sent by attacks at a volume designed to overwhelm the application server in the hopes of preventing legitimate users from accessing the application.

The opposite of a "flood" attack is a "slow" attack. In these, the same proper HTTP behavior is exploited by attackers who make requests and then very slowly receive responses. And by very slowly we're talking about 2800 baud rate slow. This causes the server's send queues to remain in use and forces the server to open new connections to serve other users until there are no connections available. This is a resource consumption attack that simply exploits a perfectly acceptable and secure web or application server to cause availability problems for legitimate users.

Now, the problem is that the behavior of HTTP does not generally fall under the purview of developers. After all, they didn't write the HTTP handlers, they rely on platform-provided, industry standard software for that. This means the responsibility for detecting and dealing with such attacks falls on the operations/network teams, upstream, where such behavior can be dealt with.

When developers choose to address these resource-consuming attacks, their best option is to use authorization-based services upstream to allow or deny these requests. Such services can be deployed as network-services or as "pre-entry" application services. The key design factor here is that the service *not* reside on the application server itself, as that defeats the solution of conserving resources by preventing requests in the first place.

Primary Responsibility: Operations / Network
Secondary Responsibility: Developers

PLATFORM LAYER

Applications, whether web or native mobile, still require logic to run on the "server-side" of the world. That means some kind of platform, e.g. Apache, is in use. When platforms are exploited it is commonly the case that it relates to how the platform handles certain protocols. As with most exploits, there are generally one of two goals behind an attack exploiting a platform vulnerability: data or denial of service. And they always target an application, which makes it a critical part of application security, particularly in the case of those driven by data exfiltration attempts.

For example, the infamous Apache Killer took advantage of how Apache mishandled an HTTP Range header that could cause excessive memory and CPU usage, ultimately resulting in a denial of service condition. More recently, Heartbleed exploited a defect in the OpenSSL library implementation of TLS/DTLS (transport layer security protocols) that resulted in the leaking of data in memory from the server to the client, and vice-versa.

These types of vulnerabilities are well outside the control of developers and, for the most part, operations as well. Platform security requires constant vigilance to determine the existence of such vulnerabilities and a subsequent plan to address. That plan is almost always the application of a hotfix from the platform vendor in question.

In cases where the platform is mishandling an HTTP header, it can be possible for developers to implement a fix if they (1) have access to the source code or (2) are able to apply such a fix before the platform processes the header. In some cases, operations may be able to configure the platform in a way that avoids processing of the header in question, or otherwise manipulates it to avoid the condition that would trigger a problem once processed. The ability of developers and operations to address these types of platform vulnerabilities depends highly on what specific pieces of a protocol are being exploited.

Network teams, in these cases, have a far better chance of preventing exploitation as they often have the tools at their disposal to inspect and filter attempted exploits upstream and thus prevent them from reaching the server in the first place. Such mitigations should be temporary, as the best response is to apply whatever patches are required when they become available. But in the meantime, mitigating upstream, in the network provides operations and developers with the time they need to resolve in a more permanent fashion.

Primary Responsibility: Network / Operations
Secondary Responsibility: Developers

This is by no means an exhaustive exploration of potential application attacks. Most application security concerns, however, fall into one of these three layers of the application security stack and thus provide a good starting point to understanding where to apply the proper mitigations and when.

LORI MACVITTIE is a subject matter expert on emerging technology across F5's entire product suite. MacVittie has extensive development and technical architecture experience in both high-tech and enterprise organizations, in addition to network and systems administration expertise. Prior to joining F5, MacVittie was an award-winning technology editor at Network Computing Magazine. She holds a B.S. in Information and Computing Science, and an M.S. in Computer Science, and is an O'Reilly author

SECURITY THREATS ARE COMING

Developers have become increasingly aware and concerned about security threats. In fact, according to this year's survey, over half of the respondents noted that developers are primarily responsible for security over frameworks and security teams. One of the first steps in building secure applications and preventing any attacks is to know what types of attacks there are.

We asked our audience what their top security threat concerns were. Drawing from over 1,000 responses, we broke down the top 7 security threats, defined them, and suggested actions to take to defend your kingdom against these attacks.

PHISHING 43%

Phishing is an attempt through email, chat, or social networks to trick users into divulging information like passwords or credit card numbers by posing as a legitimate source, such as a bank.

DEFENSE: Education around spotting phishing attempts and browser certificates are the best ways to counteract phishing.

SQL INJECTION 49%

An attack where harmful SQL statements will be entered into a data entry field, then executed.

DEFENSE: One of the most basic defenses is to use parameterized statements that create placeholders out of an entry that cannot be executed as SQL statements.

CSRF 30%

An attack that tricks an end user into executing unwanted actions on a web application in which they're currently authenticated.

DEFENSE: For an automated defense, check the standard headers to verify the request is same origin and check the CSRF token. For a manual defense, require user interaction for authorizing transactions (re-authentication, etc) or, less intrusively, use transaction IDs.

XSS 37%

Cross-site scripting is when an attacker injects client-side scripts into web applications viewed by other users, so the web site attacks its own users without the knowledge of the site owners.

DEFENSE: Using output escaping techniques to cover XML significant characters, disabling scripts, and using HTML sanitation engines can all prevent XSS attacks.

TROJANS 26%

A program that disguises itself as a trusted application, when in fact it harbors malicious code.

DEFENSE: Education—because Trojans need your permission to run on your computer, always be wary of opening files when you don't know their source.

MAN-IN-THE-MIDDLE 31%

When an attacker is secretly intercepting and potentially sabotaging communication between two parties who think that they are in direct communication with one another.

DEFENSE: There are cryptographic protocols with various forms of endpoint authentication specifically to prevent MITM attacks. Examples include public key infrastructure (like TLS), secure DNS extensions, etc.

DDoS 46%

A Distributed Denial of Service attack occurs when multiple systems flood the bandwidth of a target system, such as a web site, often coordinated by using computers infected with a virus to automatically ping the site.

DEFENSE: Using automatic traffic pattern analysis technologies to identify threats, traffic scrubbing filters, and using cloud provider solutions can help mitigate attacks.

Smart Contract Security:

How to Never Break the Blockchain

QUICK VIEW

01 Smart contracts are one of the most promising applications of blockchain technology.

02 Security in smart contracts should be the number one concern during development.

03 It's still early, and smart contract technology is evolving.

BY **LEFTERIS KARAPETSAS**
ETHEREUM CORE DEVELOPER AT **BRAINBOT AG**

This article will introduce the reader to the concept of smart contracts and how they are an essential tool for the blockchain world. The article will focus on the aspect of security around smart contracts, common pitfalls and how the user can avoid them. Finally we will see examples of security breaches in smart contracts in the past and what are the best development practices to assure security.

THE BLOCKCHAIN

The blockchain is essentially a very big database of transactions, also known as a transaction ledger. There are many variations but they all share some common properties inspired by the original Bitcoin Blockchain.

A blockchain is made up of a series of blocks. Each block holds a number of transactions that have occurred and a hash of the previous block. Thus all blocks are linked and form a chain. All the transactions in a new block are verified by solving a hard cryptographic puzzle called Proof-Of-Work by many computers around the world. These are the so-called miners.

Miners calculate the Proof-Of-Work hash of all transactions, in essence sealing the new block, and then transmit it to the network so that all nodes know a new block has been produced. All miners compete with each other in order to produce the new block first and get it accepted by the rest of the network. The incentive to do so is that the miner who produces a new block also gets a particular amount of the token of the blockchain in his account. This is how new Bitcoin, Ether, and most other cryptocurrencies are created.

SMART CONTRACTS

Smart contracts are considered one of cryptocurrency's most valuable aspects next to the transfer of value. The term was coined by Nick Szabo in 1994 in the context of bringing the practices of contractual law in the design of electronic commerce.

> *A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs.*

In practice right now the most mature platform for smart contracts is the Ethereum Blockchain. It's the second biggest blockchain in market cap and is designed to be a platform for the development and deployment of smart contracts. It uses a Turing complete virtual machine called the Ethereum Virtual Machine (EVM) which empowers users of the blockchain to executes arbitrary code in a trust-less environment. The execution of EVM code is guaranteed and deterministic. EVM code execution is quite slow, and each OPCODE has a particular associated cost which the initiator of a transaction that executes the code is going to pay. For details refer to the Ethereum Yellow Paper.

There have been a few programming languages that translate into EVM, including Mutan, LLL, and Serpent. But without doubt the most well-developed and supported language for smart contracts and the EVM is the contract-oriented language Solidity. In Solidity contracts are first-class objects and a lot of the blockchain attributes involved in EVM

transactions, such as the block timestamp or the message sender's address, are exposed to the developer for use in code. From here on in the article, whenever we refer to smart contracts, we will refer to Solidity smart contracts.

```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) {
        uint y = 10;
        storedData = x + y;
    }

    function get() constant returns (uint retVal) {
        return storedData;
    }
}
```

Above is an example smart contract written in Solidity. A contract, when deployed in the blockchain, has an address, much like a normal user account. In addition to that, it has code attached, which is the EVM translation of the contract, and also a persistent data storage. When a node executes transactions related to a contract, variables like `uint storedData` reside in Storage. They are the so called State Variables. Apart from the persistent data storage Solidity also has an ever-expanding memory where variables can be stored. The memory is wiped for each transaction and it always begins with a clean slate. Finally small local variables can also be stored on the stack.

AUTHORING SECURE SMART CONTRACTS

Smart contracts act as intermediaries between various parties, and in times may hold large amounts of money. Thus, it is only natural that security should be a primary concern. In this section we will sum up a few good rules to write secure smart contracts. Keep in mind that this list is not exhaustive and smart contract security is an evolving topic. Visit the "Further Reading" links on the bottom of the article for more information.

REENTRANCY PROTECTION

Contracts that keep any kind of ledger of their user's assets in their storage are vulnerable to re-entrancy attacks.

EXPLANATION OF THE ANTI-PATTERN

Take a look at the vulnerable contract below.

```
contract DiscountPool {
    mapping (address => uint) tokens;

    // get the amount of money corresponding to my number
of tokens
    function getMoney() payable {
        if (msg.sender.call.value(tokens[msg.sender])()) {
            tokens[msg.sender] = 0;
        }
    }

    function sendToken(address receiver, uint amount) {
        if (tokens[msg.sender] < amount) {
            throw;
            }
        tokens[msg.sender] -= amount;
        tokens[receiver] += amount;
    }
}
```

The above would be the most natural way one would write the contract in any other programming language. But unfortunately what comes naturally for most other programming languages poses a very serious threat in Solidity. Picture the following attacking contract:

```
contract DiscountPoolAttack {

    DiscountPool discountPool;
    address owner;
    function DiscountPoolAttack(address discountPoolAddress)
    {
           owner = msg.sender;
           discountPool = DiscountPool(discountPoolAddress);
    }

    // fallback function
    function () {
        discountPool.getMoney()
    }

   function getMyLoot() payable {
       if (msg.sender != owner) {
           throw;
       }
       msg.sender.send(this.balance);
   }
}
```

The nameless function above is what in Solidity is called the fallback function. Whenever there is a call/send to another contract which sends money or there is a call to a non-existing function, the callback function is also called.

What the malicious contract above achieves is that by using the `attack()` function it starts a recursive re-entrancy call chain which calls the `getMoney()` function recursively, not hitting the `tokens[msg.sender]=0` line until it is too late and the entire `DiscountPool` contract above is drained of its funds. After the deed is done, the attacker only needs to call `getMyLoot()` to receive all of his Ether from the contract.

An important thing to note here is that if send() is used instead of call() then this exploit is not possible since send() only forwards enough gas to the callee for a value transfer and if anything else is attempted then it will fail.

MITIGATION

The solution here is to utilize the Checks-Effects-Interactions pattern.

```
function getMoney() payable {
    uint mytokens = tokens[msg.sender];
    tokens[msg.sender] = 0;
    if (msg.sender.call.value(mytokens)()) {
        throw;
    }
}
```

Essentially the ledger of the user's balance is changed before everything and if the sending does not work then the entire transaction is reverted via the use of `throw;`.

EXAMPLE OF THE VULNERABILITY IN THE WILD

This is the most well-known vulnerability in smart contracts because the arguably most famous Ethereum smart contract, The DAO, was hacked because of it. On the 17th of June, an as-of-yet unknown attacker took advantage of this vulnerability in the DAO's code here and slowly extracted roughly 30% of the total value held in the contract. At the time, that Ether value was estimated to be about 50 million dollars.

UNCHECKED SENDING OF FUNDS

All functions that send money like `send()` and `call()` should always be checked for success or failure.

EXPLANATION OF THE PROBLEM

Sending money does not always succeed. The user may run out of gas or the transaction may fail for some other reason. That's why the result of money transfers should always be checked.

```
// Create a new custodian of funds and transfer
// the funds to him
function newCustodian(address custodian_address) {
    custodian_address.send(funds);
    custodian = custodian_address;
}
```

In the above example, the `send()` may fail and we will miss it. When this happens our contract will be broken, attributing the `custodian` title (whatever that may be) to someone who has not actually received the funds.

MITIGATION

Always wrap `send()` and `call()` within `if` statements.

```
// Create a new custodian of funds and transfer
// the funds to him
function newCustodian(address custodian_address) {
    if (custodian_address.send(funds)) {
        custodian = custodian_address;
    }
}
```

A good thing about the solidity compiler is that it will issue warnings whenever it sees that you have unchecked sends. Solidity warnings are serious and should always be taken care of.

EXAMPLE OF THE VULNERABILITY IN THE WILD

An older version (0.4.0) of a Game Of Thrones Pyramid contract, the King of the Ether Throne, was disrupted by a non-checked `send()` that failed here.

```
if (currentMonarch.etherAddress != wizardAddress) {
    currentMonarch.etherAddress.send(compensation);
} else {
    // When the throne is vacant, the fee accumulates
    // for the wizard.
}

// Usurp the current monarch, replacing them
// with the new one.
pastMonarchs.push(currentMonarch);
currentMonarch = Monarch(
    msg.sender,
    name,
    valuePaid,
    block.timestamp
);
```

Newer versions of that contract are not affected.

USING BLOCK DATA FOR RANDOMNESS

In the EVM, code execution is deterministic. Thus it is very tricky to attempt to generate random numbers.

EXPLANATION OF THE PROBLEM

The hash or the timestamp of a block in the future is unpredictable, deterministic, and is going to be the same for everyone when a particular block N is mined. For that reason, some contracts may choose to use this as a source of randomness.

Unfortunately, as it turns out, this can be gamed. In creating new blocks for the blockchain, miners can control which transactions make it into a block and in which order. Thus they can try to manipulate the outcome of such random numbers as shown here and here.

MITIGATION

Random numbers on the blockchain remain quite a hard problem. There are, though, some more involved solutions that are harder to game. Two such examples in Ethereum are the Maker Darts and Randao.

They both function in a similar way. Users make timed deposits accompanied by a secret. A random number is generated every so often by the contract using a combination of the user secrets. If participants lie about their secrets they lose their deposits.

EXAMPLE OF VULNERABILITY IN THE WILD

One example of a contract generating random numbers by using block data can be seen in this Roulette game.

UNBOUND LOOPS

Every block in the Ethereum blockchain has a gas limit. Each operation of the EVM costs a certain amount of gas. If some operation you attempt consumes so much gas as to hit the block gas limit then it will fail and not be included

in the block. This can happen in many different ways, but one that all developers should be aware of is iterating over storage variables.

EXPLANATION OF THE PROBLEM

Imagine we have a contract that keeps a registry of addresses to names and also wants to at some point pay 1 Wei (smallest ether denomination) to all users.

```
contract UserRegistry {
    address[] users;
    mapping (address => string) addressMap;
    mapping (string => address) nameMap;


    function register(string name) {
        if (sha3(addressMap[msg.sender]) == sha3('')) {
            throw;
        }
        addressMap[msg.sender] = name;
        nameMap[name] = msg.sender;

        users.push(msg.sender);
    }

    function payAllUsers() {
        for (uint i = 0; i < users.length; i++) {
            if (!users[i].send(1)) {
                throw;
            }
        }
    }
}
```

It all looks fine at first, but if you take a close look at the `payAllUsers()` function you will see a loop that is only bound by the size of a storage array. That is very dangerous! The more the registry grows, the more gas the function call will be costing. Eventually it will reach the block gas limit and then the contract will become unusable.

As a bonus problem here, any malicious user can also have a contract whose fallback function `throws`. If someone does that then our loop will always hit that malicious fallback function when iterating and fail the loop every time.

MITIGATION

Never use loops for something that depends purely on a storage variable and that will grow linearly with time. There should either be an upper bound calculated to fit within the maximum gas limit or a way to split the loop into multiple transactions.

EXAMPLE OF VULNERABILITY IN THE WILD

A betting dice game called Etherdice was stopped in its tracks due to this vulnerability. It was iterating all of the bets in the same loop and when the number of bets reached the gas limit the contract stopped functioning.

COMPILER BUGS

All deployed Solidity contracts are compiled by the Solidity compiler. The compiler may have a bug and a big batch of contracts may suddenly find themselves vulnerable. Anything that may go wrong, will go wrong. This is a rule to live by in smart contracts.

MITIGATION

Always have some form of a backup plan in your contracts. Things that you never expected to happen may go wrong, or your contract may be vulnerable to a bug in the compiler discovered a long time after you deploy.

```
function escapeHatch() {
    if (msg.sender == owner && redAlertCheck()) {
        msg.sender.send(this.balance);
    }
}
```

A function like the above would allow the maintainer of a contract to cancel everything and get all of the funds out of a vulnerable contract. `redAlertCheck()` is optional and could be a check for some invariants of you contract depending on its functionality.

Always remember the DAO. If the DAO had an escape hatch all the funds could have been saved.

EXAMPLE OF VULNERABILITY IN THE WILD

On November 1, 2016, a compiler bug that could have affected all contracts deployed with Solidity was discovered. Taking advantage of a compiler bug, a malicious attacker could use an overflow bug and rewrite storage variables. The bug was fixed immediately, and we were quite lucky this time because very few contracts were actually affected and had to be redeployed.

FINAL WORDS

Ethereum and smart contracts as a platform are both still in their infancy. The tools at our disposal are maturing, and developers are becoming more security-aware. Soon we will also have formal verification of Solidity contracts, proof that parts of your code fulfill a certain mathematical specification.

As a contract developer, you should adhere to the guidelines described here, monitor Ethereum-related media for news regarding vulnerabilities, and always include failsafes such as escape hatches in your code.

FURTHER READING

- Ethereum Yellow Paper
- Solidity Documentation
- A Survey Of Attacks on Ethereum Smart Contracts
- Breaking Ethereum

LEFTERIS KARAPETSAS is a Berlin-based Developer and University of Tokyo graduate, Lefteris is an Ethereum developer and Emacs user.

Setting the Standard for Application Security

Speed

Fast results combined with unmatched breadth, depth, and accuracy of analysis.

Trust

Our tools are widely adopted by developers, the security community, and standards groups.

Insight

A unified view of overall software health that supports total risk visibility.

Flexibility

Our solutions are designed to scale as your business needs change.

SYNOPSYS®

Visit www.synopsys.com/software or call 1(800) 541-7737 to learn more.

Synopsys is Setting the Standard for How Companies Create and Secure Software

Synopsys provides tools that are the standard for software development, testing, and procurement processes in 19 of the top 25 software companies, 11 out of the top 15 Automotive OEMs, and in organizations across many different industries. Our end-to- end offering gives companies a turnkey, scalable approach for minimizing software-related business risks, maximizing release predictability and speed, and ensuring internal and external standards compliance.

We're also involved with software standards bodies (FDA, SAE, and UL, just to name a few), helping to define next generation software quality and security testing. Today, "good enough" just isn't good enough anymore, especially when it comes with the daunting prospect of recalls, updates, and emergency patches. Our fast and accurate platform of testing tools fits within your existing Software Development Life Cycle, works well with Agile methodologies, and is embraced by developers worldwide.

> Our end-to-end offering gives companies a turnkey, scalable approach to ensure internal and external standards compliance.

And given that up to 90% of software consists of code obtained through third parties (the cyber supply chain), our Synopsys Software Integrity Platform identifies third-party components and their vulnerabilities, manages open source licenses and copyright issues, and secures the intellectual property inside your applications and firmware.

WRITTEN BY ROBERT VAMOSI
CISSP, SECURITY STRATEGIST, **SYNOPSYS**

PARTNER SPOTLIGHT

Synopsys

Synopsys, Inc. provides the solutions needed to deliver smart, secure products for the era of connected everything.

CATEGORY

Application Security

NEW RELEASES

Synopsys Releases Coverity 8.5 Static Analysis Tool

Synopsys Releases Seeker 3.8 Runtime Security Analysis Tool for Web Applications

CASE STUDY

Parkeon is a key player in the urban mobility sector and a global provider of parking and transport management solutions. Parkeon offers a unique range of parking control and payment services in 55 countries and more than 3,000 cities around the world. While using Synopsys Seeker, Parkeon has identified three key benefits that demonstrate that it is the tool for them. First, Seeker ensures that the entire system, end to end, complies with security standards such as PCI-DSS by understanding how data flows throughout the entire application. It identifies vulnerabilities in relation to their impact on sensitive data. Second, Seeker facilitates communication between the test and development teams by linking vulnerabilities back to the offending source code. Unlike other dynamic testing tools which report vulnerabilities by the offending URL, Seeker automatically ties those vulnerabilities back to the source code where the fix needs to be applied. And third, Seeker improves security awareness and helps train developers for more secure coding practices. Parkeon's developers and testers are trained on the basis of OWASP TOP10, but they are not information security experts. By providing a replay of every attack, explaining the business risks and providing relevant remediation suggestions, Seeker helps their test and development teams to acquire awareness and training in an ongoing manner, thus improving the security of their code.

STRENGTHS

- Automated Security Testing for Agility
- Advanced Protection of Sensitive Data
- Unmatched Accuracy
- Clear Paths to Remediation

NOTABLE CUSTOMERS

- Alcatel Lucent
- Bally Technologies
- Direct Edge
- Parkeon
- AirFrance
- S2E
- MStar

BLOG blogs.synopsys.com/codereview

TWITTER @SW_Integrity

WEBSITE synopsys.com

Executive Insights on Application and Data Security

BY **TOM SMITH**
RESEARCH ANALYST AT **DZONE**

QUICK VIEW

01 5 keys to security: know the fundamentals, execute best practices, integrate security into the SDLC, practice data-centricity, and test and monitor continuously.

02 The security landscape has changed to involve more threat access points, and the number of threats and hacks expanding daily.

03 The most effective security techniques combine different approaches and tools throughout the SDLC process, of which testing is integral.

To gather insights on the state of application and data security, we spoke with 18 executives who are involved in application and data security for their clients. Here's who we talked to:

SAM REHMAN, CTO, Arxan
BRIAN HANRAHAN, Product Manager, Avecto
PHILIPP SCHÖNE, Product Manager IAM & API, Axway
BILL LEDINGHAM, CTO, Black Duck
AMIT ASHBEL, Marketing, Checkmarx
JEFF WILLIAMS, CTO and Co-Founder, Contrast Security
TZACH KAUFMAN, CTO and Founder, Covertix
JONATHAN LACOUR, V.P. of Cloud, Dreamhost
ANDERS WALLGREN, CTO, Electric Cloud
ALEXANDER POLYKOV, CTO and Co-Founder, ERPScan
DAN DINNAR, CEO, HexaTier
ALEXEY GRUBAUER, CIO, Jumio
JOHN RIGNEY, CTO, Point3 Security
BOB BRODIE, Partner, SUMOHeavy
JIM HIETALA, V.P. Business Development Security, The Open Group
CHRIS GERVAIS, V.P. Engineering, Threat Stack
PETER SALAMANCA, V.P. of Infrastructure, TriCore Solutions
JAMES E. LEE, EVP and CMO, Waratek

KEY FINDINGS

01 The most important elements of application and data security are: **1) focusing on the fundamentals; 2) identifying best practices, frameworks, and architectures; 3) embedding security in the entire software development lifecycle (SDLC); 4) being data-centric; and 5) testing and monitoring continuously.** The four pillars of security are 1) securing the database to prevent SQL injection; 2) scanning software for sensitive data discovery; 3) actively monitoring the app and the database; and 4) providing dynamic data masking as needed. There is a huge variance in best practices from one manufacturer and operating system to another. 99% of application developers will benefit from application frameworks. Moving security to the left of the SDLC inherently provides visibility across the entire process, providing much-needed insight for developers, engineers, and security professionals.

02 The programming languages and frameworks most frequently mentioned by respondents were **JavaScript, Java, and C++**; however, there were mentions of 28 others along with a couple of companies using 20 and 70 additional languages respectively and two companies using whatever their clients are using.

03 The cybersecurity landscape continues to evolve with **more threats and more access points** thanks to IoT and connections to the cloud. With all of these access points and connections, vulnerabilities and hacks will continue to grow. The focus of the attacks has changed from users, credit cards, and malware to industry specific vectors like oil and gas and retail. Hackers are going after personal identifiable information (PII) for identity theft like passports and social security numbers. The old threats have been automated. Countries and businesses are providing root kits and services to other hackers because there is so much money to be made by the hackers and the companies providing the tools.

04 The most effective security techniques and tools are: **1) a combination of different approaches; 2) a secure SDLC process**

with which testing is integral; and, 3) security baked into the architecture. Companies are identifying security issues by layering static, dynamic, and interactive tests. Fundamental security elements include encryption when data is in transit, at rest, and flowing between data centers. Platforms can serve as the foundational element for authentication, authorization, and other techniques to ensure a strong foundation.

05 **Most real-world problems are being solved in financial services and healthcare** since these are the most highly regulated industries. Solutions revolve around following the best practices like PCI implementation and the OWASP 10. However, there are a lot of companies who are not in highly regulated industries who are putting themselves, and their customers' PII, at risk. Stay current with patches and updates. Reduce mean-time-to-failure with recovery, remediation, and application of the process all taken into consideration.

06 The most common issue our respondents see affecting application and data security is **not taking a holistic view of security as a strategic necessity** as evidenced by lack of knowledge of fundamentals and best practices. Also, companies continue to spend 95% of their security budget on infrastructure and web security versus application security where 90% of attacks are aimed. In addition, companies do not have the security personnel necessary to monitor security and address vulnerabilities and concerns.

07 Virtually all of the respondents have concerns regarding the current state of application and data security. **It's bad and it's going to get worse before it gets better for a number of reasons.** According to Verizon, there are an average of 22.4 serious vulnerabilities in each application they tested. The development of IoT devices are way ahead of processes and best practices needed to create even more secure devices and applications. Too many organizations are not looking at the research and strategies as a first step before buying a tactical solution. The bad guys continue to find vulnerabilities faster than the good guys can fix the problems with nothing meaningful to disrupt the cycle. Government agencies have become quite sophisticated, but so have the bad guys. It's an ongoing "cat and mouse" game with very real implications.

08 The future of application and data security is **automation and algorithms driving artificial intelligence and machine learning**. However, we still need organizations to start looking at security as part of their SDLC and IT strategy and funding it at a sufficient level so the vision can be realized. We will use instrumentation to improve security by orders of magnitude. In the future all software will be instrumented for security all of the time. We will have insight into how an app is operating and automatically take action as a result—automated remediation.

09 Developers need to **think security first and learn and follow best practices for greater career success and longevity**. Get application security training so you know how to build resilient applications. Be aware of threat and design principles, as well as the OWASP 10 and vulnerability databases. Have a good knowledge of frameworks and the security strengths and weaknesses of the frameworks you are using. Monitor the performance of your applications and be aware of unusual or unintended use. There are greater career opportunities for developers and architects with secure coding skills. More skills equate to greater career growth. There are a lot of advantages to being a secure developer and a lot of tools available to understand and learn secure coding.

10 Additional considerations regarding application and data security include:

- **Software security is invisible**. In the marketplace, you get the same price for software regardless of how secure it is. There's no incentive to build secure software. This has to change if we hope to address the problem.
- **Applications are currently working in silos**. Ultimately we'll connect all applications in a safe and secure way just by using a browser. Apps will communicate using agreed upon security protocols.
- There's not a lot of emphasis on protecting data. **How do we secure the data, encrypt access, and scan to know what data resides where?**
- We need to **be aware of how changes to applications threaten the service** and what new vectors of attack will become popular.
- **Privacy is huge**. Credit card data is one thing; however, heart rate monitors, knowing when you're at home or not, and where your car is parked, and for how long, have massive privacy implications that affect peace of mind.
- As we become more agile and cloud based, there are more challenges with changed in cybersecurity, as well as security and development in the cloud. **How does compliance, security, and operations address these challenges?**
- **How is DevOps affecting the security process?** Does it help or hurt? What does it take to make the transition to Dev/Sec/Ops that enables continuous integration and secure integration? Communications between developers, security, and operations becomes critical.
- **How realistic is it to expect developers to write more secure code?** Is over-reliance on the status quo delaying the development and implementation of new security technologies and techniques?
- **Use smaller chunks of data** because it's worth less to hackers.
- **Blockchain is not fine, it's chaos**.
- **How do we go faster securely?** Make security and accelerator to increase speed to market.

TOM SMITH is a Research Analyst at DZone who excels at gathering insights from analytics—both quantitative and qualitative—to drive business results. His passion is sharing information of value to help people succeed. In his spare time, you can find him either eating at Chipotle or working out at the gym.

Diving Deeper

INTO SECURITY

TOP #SECURITY TWITTER FEEDS

To follow right away

 @briankrebs

 @jeremiahg

 @MME_IT

 @garyjdavis

 @troyhunt

 @tojarrett

 @adamcaudill

 @WeldPond

 @unix_root

 @cybersecboardrm

TOP SECURITY REFCARDZ

MQTT

dzone.com/refcardz/getting-started-with-mqtt

Explores the fundamentals of MQTT, including message types, QoS levels, and, of course, security.

Getting Started With Industrial Internet

dzone.com/refcardz/getting-started-with-industrial-internet

Covers the basics like sensors and actuators, industrial control systems, human-machine interfaces, real-time streaming data, device security, and more.

Spring Security 3

dzone.com/refcardz/expression-based-authorization

Begin to master Spring Security by learning the key features of expression-based authorization for a challenging framework.

TOP SECURITY WEBSITES

Krebs on Security krebsonsecurity.com

Schneier on Security schneier.com

Troy Hunt troyhunt.com

SECURITY ZONES

Learn more & engage your peers in our Security-related topic portals

Cloud

dzone.com/cloud

Cloud Zone covers the host of providers and utilities that make cloud computing possible and push the limits (and savings) with which we can deploy, store, and host applications in a exible, elastic manner. This Zone focuses on PaaS, infrastructures, security, scalability, and hosting servers.

DevOps

dzone.com/devops

DevOps is a cultural movement, supported by exciting new tools, that is aimed at encouraging close cooperation within cross-disciplinary teams of developers and IT operations/ system admins. The DevOps Zone is your hot spot for news and resources about Continuous Delivery, Puppet, Chef, Jenkins, and much more.

Performance

dzone.com/performance

Scalability and optimization are constant concerns for the developer and operations manager. The Performance Zone focuses on all things performance, covering everything from database optimization to garbage collection, tool and technique comparisons, and tweaks to keep your code as efficient as possible.

SECURITY NEWSLETTERS

SANS Newsbites

sans.org/newsletters/newsbites

Security Magazine eNewsletters

securitymagazine.com/eNewsletters

CSO Security Newsletters

csoonline.com/newsletters/signup.html

Solutions Directory

This directory contains anti-tamper software, authentication, Cloud access security, DDoS protection, endpoint security, and penetration testing tools, as well as many other tools to assist your application security. It provides free trial data and product category information gathered from vendor websites and project pages. Solutions are selected for inclusion based on several impartial criteria, including solution maturity, technical innovativeness, relevance, and data availability.

PRODUCT	COMPANY	CATEGORY	FREE TRIAL	WEBSITE
Acunetix Web Vulnerability Scanner	Acunetix	DAST, IAST	14 Day Free Trial	acunetix.com/vulnerability-scanner
Adallom	Microsoft	Cloud Access Security Broker	Demo Available by Request	microsoft.com/en-us/cloud-platform/cloud-app-security
Advanced Authentication	CA Technologies	Authentication	On Request	ca.com/us/products/ca-advanced-authentication.html
Airlock Suite by Ergon Informatik	Ergon Informatik AG	WAF, Authentication, Identity Management	Demo Available by Request	airlock.com/en/home
Akamai	Akamai	CDN, DDoS Protection, WAF	N/A	akamai.com
Alert Logic Security-as-a-Service	Alert Logic Inc.	Intrusion Prevention System, Cloud Access Security Broker, WAF	Available by Request	alertlogic.com/solutions
Amazon WAF	Amazon	WAF	N/A	aws.amazon.com/waf
AppMobi Security Kit	AppMobi	Apache Cordova App Encryption and Authentication	Available by Request	appmobi.com/security-kit-trial
AppSpider Pro by Rapid7	Rapid7	DAST	Demo Available by Request	rapid7.com/products/appspider
Appthority	Appthroity	Mobile AST	Available by Request	appthority.com/.com
AppWall by Radware	Radware	WAF, DDoS Protection	Available by Request	radware.com/products/appwall
Arbor Networks APS	Arbor Networks	DDoS Protection	N/A	arbornetworks.com/ddos-protection-products/arbor-aps
Armor Complete	Armor Defense Inc.	Cloud Security Platform	Available by Request	armor.com/security-solutions/armor-complete
Arxan Application Protection	Arxan	Anti-Tamper Software	Demo Available by Request	arxan.com/products/product-overview
AuditMyApps by Pradeo	Pradeo	Mobile AST	Available by Request	pradeo.com/en-US/apps-security-test.com
Auth0	Auth0	Authentication	Free Tier Available	auth0.com/how-it-works
Authentification Management Platforms	Gemalto	Authentication	Demo Available by Request	gemalto.com/enterprise-security/identity-access-management
Barracuda Firewal	Barracuda Networks	WAF	N/A	barracuda.com/products/ngfirewall

PRODUCT	COMPANY	CATEGORY	FREE TRIAL	WEBSITE
BeEF	Browser Exploitation Framework Project	Penetration Testing	Open Source	beefproject.com
Bit9 + Carbon Black	Carbon Black	Endpoint Security	Demo Available by Request	carbonblack.com
Black Duck Hub	BlackDuck	Open Source Scanning	Demo Available by Request	blackducksoftware.com/products/hub
BladeLogic Threat Director	BMC (Sponsor)	Penetration Testing	On Request	bmc.com/it-solutions/bladelogic-threat-director.html
Blue Coat Cloud	Blue Coat Systems Inc.	Cloud Access Security Broker, WAF	Available by Request	bluecoat.com/products-and-solutions/cloud-delivered-web-security-services
Bluebox	IBM Bluemix	Mobile Access Security Broker	Demo Available by Request	blueboxcloud.com
Breakpoint/VE	Ixia	Penetration Testing	Demo Available by Request	ixiacom.com/products/breakingpoint-ve
BrightCloud Threat Intelligence by Webroot	Webroot	DAST	N/A	webroot.com/us/en/business/threat-intelligence
Burp Suite by PortSwigger	PortSwigger LLC	SAST, DAST, Penetration Testing	Free Tier	portswigger.net/burp
CD Protection by CD Networks	CDNetworks	CDN, WAF, DDoS Protection	N/A	cdnetworks.com/products/cloud-security/ddos-protection
Checkmarx CxSAST	Atlassian	SAST, DAST, RASP	Available by Request	checkmarx.atlassian.net/wiki/display/KC/Checkmarx+CxSAST+Overview
Cigital	Cigital, Inc.	SAST, DAST	N/A	cigital.com
CipherCloud	CipherCloud	Cloud Access Security Broker	Available by Request	ciphercloud.com
Cisco ACE WAF	Cisco	WAF	N/A	bit.ly/2g3u761
Cloud Endpoint Protection	CSC	Endpoint Security	Demo Available by Request	bit.ly/2g4XZ3q
CloudFlare	Cloudfare, Inc.	CDN, DDoS Protection, WAF	N/A	cloudflare.com/
CloudFront by Amazon	Amazon	CDN, DDoS Protection	N/A	aws.amazon.com/cloudfront
CloudLock Security Fabric	Cisco	Cloud Access Security Broker	Demo Available by Request	cloudlock.com/platform/fabric-services
Cloudmark Trident	Cloudmark	Real-Time Threat Detection	On Request	cloudmark.com/en/s/products/cloudmark-trident
CloudPassage Halo	CloudPassage	Cloud Access Security Broker	Demo Available by Request	cloudpassage.com/products
CloudSOC by Elastica	Elastica	Cloud Security Testing/Scanning	Free Risk Assessment	elastica.net/cloudsoc
CodeProfiler by Virtual Forge	Virtual Forge GmbH	SAST	Available by Request	virtualforge.com/en/portfolio/codeprofiler.html
CodeSonar	GrammaTech	SAST	30 Day Free Trial	grammatech.com/products/codesonar
ContextIntelligence by Yottaa	Yottaa	CDN, DDoS Protection, WAF	N/A	yottaa.com/product
Contrast Enterprise	Contrast Security	IAST, RASP	Demo Available by Request	contrastsecurity.com/whats-included
Covata Platform	Covata	Authentication	Demo Available by Request	covata.com/solutions/covata-platform

PRODUCT	COMPANY	CATEGORY	FREE TRIAL	WEBSITE
Coverity	Synopsys/Coverity (Sponsor)	Static Code Analysis	On Request	synopsys.com/software-integrity/products/static-code-analysis.html
Datavantage	Varonis	Insider Threat Detection/ Prevention	Available by Request	varonis.com/products/datadvantage
Datiphy Platform	Datiphy	Real-Time Threat Detection	Demo Available by Request	datiphy.com/products/platform
DDoS Strike by Security Compass	Security Compass	DDoS Protection	Available by Request	securitycompass.com/dds
Deepfield Defender	Deepfield	DDoS Protection	Request Demo	deepfield.com/products/deepfield-defender
Defendpoint by Avecto	Avecto	Endpoint Security	Available by Request	avecto.com/defendpoint
DenyAll WAF	DenyAll	WAF	N/A	denyall.com
Discover by UpGuard	UpGuard	Penetration Testing, Analytics	Demo Available by Request	upguard.com/discover
Evident.io	Evident.io	Security and Compliance Automation	Demo Available by Request	evident.io/what-is-esp
F5 Big-IP ADC platform	F5 Networks Inc.	WAF, DDoS Protection	N/A	f5.com/products/big-ip
Falcon Host by CrowdStrike	CrowdStrike	Endpoint Security	Available by Request	crowdstrike.com/products/falcon-host
FireEye NX	FireEye, Inc.	Web Server Scanner, WAF	N/A	fireeye.com/products/nx-network-security-products.html
Fortigate Firewall Platform by Fortinet	Fortinet, Inc.	WAF	Available by Request	fortinet.com/products-services/products/firewall.html
FortiWeb by Fortinet	Fortinet, Inc.	WAF	Available by Request	fortinet.com/products-services/products/web-application-firewall/fortiweb.html
Forum Sentry	Forum Systems	Authentication	Demo Available by Request	forumsys.com/en/products/forum-sentry-api-security-gateway
HP Fortify Static Code Analyzer	Hewlett Packard Enterprise	SAST, DAST, IAST, RASP	Available by Request	hp.com/us/en/software-solutions/static-code-analysis-sast
Imperva Incapsula	Imperva	WAF, DDoS Protection	N/A	imperva.com/Products/ImpervaIncapsula
InfoBlox DNS Firewall	Infoblox	WAF	60 Day Free Trial	infoblox.com/products/dns-firewall
Intelligent Next-Gen T-Series Firewall by Hillstone Networks	Hillstone Networks	WAF	N/A	hillstonenet.com/our-products/intelligent-next-gen-firewalls-t-series
Kali Linux	Kali Linux	Penetration Testing	Open Source	kali.org
Klocwork by Rogue Wave Software	Rogue Wave Software	Code Quality Scanning	Available by Request	klocwork.com
Kona Site Defender by Akamai	Akamai	WAF, DDoS Protection	N/A	akamai.com/us/en/solutions/products/cloud-security/kona-site-defender.jsp
Level 3 Content Delivery Network	Level 3	CDN, DDoS Protection	N/A	level3.com/en/products/content-delivery-network
LogRhythm Security Intelligence Platform	LogRhythm, Inc.	Predictive Security Analytics	Demo Available by Request	logrhythm.com/products/security-intelligence-platform
Malwarebytes Endpoint Security	Malwarebytes	Endpoint Security	N/A	malwarebytes.com/business/endpointsecurity
Metafender by OPSWAT	OPSWAT	SAST	Available by Request	opswat.com/metadefender-core

PRODUCT	COMPANY	CATEGORY	FREE TRIAL	WEBSITE
MetaFlows	Metaflows, Inc.	Cloud Security Scanning	14 Day Free Trial	metaflows.com
Metasploit by Rapid7	Rapid7	Penetration Testing	Open Source	rapid7.com/products/metasploit
ModSecurity	Trustwave Holdings Inc.	WAF	Open Source	modsecurity.org
N-Stalker Cloud Web Scan	N-Stalker	SAST, DAST	Free Tier Available	nstalker.com
NetScaler AppFirewall by Citrix	Citrix	WAF	N/A	citrix.com/products/netscaler-appfirewall
Neustar	Neustar, Inc.	DDoS Protection	N/A	neustar.biz/
Nevis Security and Compliance Suite by AdNovum	Ad Novum	WAF, Authentication, Identity Management	Available by Request	adnovum.sg/en/sg/solutions/products/nevis.html
NGINX Plus	NGINX (Sponsor)	Authentication	30 Day Free Trial	nginx.com/products
Nikto2	CIRT.net	Web Server Scanner	Open Source	cirt.net/Nikto2
Nmap	NMAP.org	Penetration Testing and Network Mapping	Open Source	nmap.org
NSFOCUS Web Application Firewall	NSFOCUS	DAST, WAF	N/A	nsfocusglobal.com/waf-series
Okta Platform	Okta	Authentication	Developer Edition Free	okta.com/products/developer
Open Source Security	WhiteSource	Open Source Scanning	Trial Available	whitesourcesoftware.com/open-source-security
OWASP Zed Attack Proxy (ZAP)	OWASP	Penetration Testing	Open Source	owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
PA-7000 Series Firewall by Palo Alto Networks	Palo Alto Networks, Inc.	WAF	N/A	paloaltonetworks.com/products/secure-the-network/next-generation-firewall/pa-7000-series
Palo Alto Enterprise Security Platform	Palo Alto Networks, Inc.	RASP WAF	Available by Request	paloaltonetworks.com/products/designing-for-prevention/security-platform
Peach Fuzzer	PeachFuzzer	Penetration Testing	Available by Request	peachfuzzer.com
Prevoty	Prevoty	RASP	Demo Available on Request	prevoty.com
ProAccel by Bricata	Bricata LLC	Intrusion Prevention System	Available by Request	bricata.com/Products/About-ProAccel.aspx
ProtectWise Cloud Enterprise Security Platform	Protectwise, Inc.	CDN, App Security Scanning	Demo Available on Request	protectwise.com/platform.html
Qualys Security & Compliance Suite	Qualys, Inc.	DAST, WAF	Available by Request	qualys.com/forms/trials/suite
Risk Fabric by Bay Dynamics	Bay Dynamics	Predictive Security Analytics	Available by Request	baydynamics.com/risk-fabric
RSA ECAT	DellEMC	DAST	Available by Request	canada.emc.com/security/rsa-ecat.htm
Secure Code Warrior	SecureCodeWarrior	Gamified Training	Free Trial Available	new-www.securecodewarrior.com/developer
SecureSurf	AppRiver	WAF	30 Day Free Trial	appriver.com/services/web-protection
Security AppScan	IBM	SAST, DAST, IAST	Available by Request	03.ibm.com/software/products/en/appscan

PRODUCT	COMPANY	CATEGORY	FREE TRIAL	WEBSITE
Securonix Platform	Securonix	Real-Time Threat Detection	Demo Available by Request	securonix.com/security-intelligence
Signal Sciences Dashboard	Signal Sciences	WAF, Real-Time Threat Detection	Demo Available by Request	signalsciences.com/product
SiteLock TrueCode SAST	SiteLock	SAST, DAST	Available by Request	sitelock.com/truecode.php
Social Login	Janrain	Identity Management and Authentication	Basic is Free	janrain.com/product/social-login
Sophos Next-Gen Firewall	Sophos	WAF	30 Day Free Trial	sophos.com/en-us/products/next-gen-firewall.aspx
SRX Series Firewall by Juniper Networks	Juniper Networks	WAF	N/A	juniper.net/us/en/products-services/security/srx-series
Sucuri	Sucuri	WAF	N/A	sucuri.net
Sucuri Website Firewall	Sucuri	WAF, DDoS Protection, App Security Scanning	Available by Request	sucuri.net/website-firewall
Symantec Advanced Threat Protection	Symantec Corporation	IAST, RASP	60 Day Free Trial	symantec.com/products/threat-protection/advanced-threat-protection
Tanium Endpoint Platform	Tanium, Inc.	Endpoint Security, App Security Scanning	Available by Request	tanium.com/products
Thunder TPS by A10 Networks	A 10 Networks	DDoS Protection	N/A	a10networks.com/products/thunder-series/ddos-detection-protection-mitigation
Trend Micro Deep Security Platform	Trend Micro Incorporated	SAST, DAST	N/A	trendmicro.com/us/enterprise/cloud-solutions/deep-security
Tripwire Enterprise	Tripwire, Inc.	IAST, RASP	Demo Available on Request	tripwire.com/it-security-software/scm/tripwire-enterprise
Trustwave Secure Email Gateway	Trustwave Holdings Inc.	CDN, DAST	N/A	trustwave.com/Products/Content-Security/Secure-Email-Gateway/
Trustwave Web Application Firewall	Trustwave Holdings Inc.	WAF, Penetration Testing	N/A	trustwave.com/Products/Application-Security/Web-Application-Firewall
Vera Platform	Vera	Information Rights Managament	Available by Request	vera.com/product/sdk-api
Veracode Cloud Platform	Veracode	SAST, DAST, Mobile AST, Penetration Testing	Demo Available on Request	veracode.com/products/application-security-platform
Vormetric Transparent Encryption	Vormetric	Application Encryption	Demo Available by Request	vormetric.com/products/transparent-encryption
vSentry by Bromium	Bromium	Endpoint Security	Demo Available by Request	bromium.com/advanced-endpoint-security/protect.html
vThreat Platform	vThreat, Inc.	Penetration Testing, App Security Scanning	Available by Request	vthreat.com
WhiteHat Sentinel	WhiteHat Security	DAST	30 Day Free Trial	whitehatsec.com/products/dynamic-application-security-testing
WhiteHat Sentinel	WhiteHat Security	SAST	30 Day Free Trial	whitehatsec.com/products/static-application-security-testing
Wireshark	Wireshark Foundation	Penetration Testing and Packet-level Monitoring	Open Source	wireshark.org
Ziften	Ziften Technologies, Inc.	Endpoint Security	30 Day Free Trial	ziften.com/product-overview

GLOSSARY

AUTOMATED REMEDIATION Automatic action taken as a result of insights into how an application is operating.

BITCOIN A digital currency (cryptocurrency) that is not ruled by any governing body.

BLOCKCHAIN Essentially, a very big database of transactions, also known as a transaction ledger.

CRYPTOCURRENCY An encrypted digital exchange whose encryption techniques are used as a method to ensure that secure transactions take place that are both regulated and verified.

DATA EXFILTRATION An unauthorized transfer of data. It can be carried out manually or through a malicious automated program.

DECENTRALIZED AUTONOMOUS ORGANIZATION (DAO) An organization that serves as a form of a venture capital fund. It runs through smart contracts and its transaction records are maintained in a blockchain.

DENIAL OF SERVICE ATTACK (DDOS) A type of attack that shuts down services, usually by sending a number of requests to the service that the service cannot handle, interuppting legitimate requests of the service.

DYNAMIC APPLICATIONS SECURITY TESTING (DAST) An analysis of an application's security that only monitors the runtime environment and the code that is executed in it. It simulates potential attacks and analyzes the results.

ENCRYPTION A method for encoding data so that it is unreadable to parties without a method for decryption.

INJECTION ATTACK A scenario where attackers relay malicious code through an application to another system for malicious manipulation of the application. These attacks can target an operating system via system calls, external programs via shell commands, or databases via query language (SQL) injection.

MINERS Calculate the Proof-Of-Work hash of all transactions in a blockchain block, in essence sealing the new block and then transmiting it to the network so that all nodes know a new block has been produced.

OPEN WEB APPLICATION SECURITY PROJECT (OWASP) An online community of corporations, educational organizations, and individuals focused on providing web security tools, resources, events, and more for the wider development community.

PROTOCOL EXPLOITATION A security vulnerability that disrupts the interactions between multiple communication protocols.

RUNTIME APPLICATION SELF-PROTECTION (RASP) A feature that is built into an application in order to detect and halt attacks in real-time, automatically.

REENTRANCY ATTACKS An attack where untrusted code reenters a contract and manipulates state.

SMART CONTRACTS A computerized transaction protocol that executes the terms of a contract.

STATIC APPLICATION SECURITY TESTING (SAST) An analysis of an application's security that looks at an application's source code, bytecode, or binary code to determine if there are parts that could allow security exploits by attackers.

TURING complete A system theoretically capable of solving any computational problem if memory or runtime limitations are not taken into consideration.

WEB APPLICATION FIREWALL (WAF) An HTTP/S firewall for web applications; legacy WAFs can create network architecture complexity and aren't very accurate.