DZone

DZONE TREND REPORT

DECEMBER 2022

Enterprise Application Security

Building Secure and Resilient Applications

BROUGHT TO YOU IN PARTNERSHIP WITH

Welcome Letter

By Melissa Habit, Senior Publications Manager at DZone

Imagine yourself in Venice, where you're seated in one of Italy's most famous opera houses, *Teatro La Fenice*. The stage is illuminated by warm lights that dance along a violinist's horsehair bow as they draw the opening note in E major of Antonio Vivaldi's "Le quattro stagioni."

Vivaldi's "The Four Seasons" is recognized as an organizational masterpiece that, at the time of its conception, was an innovative approach to musical composition: a group of four violin concerti ascribing melodic expression to every season of the year.

Each concerto depicts an individual story that builds upon the next, contributing to the overall narrative. Leading a performance in this style to create a seamless work of art requires attention to detail, collaboration, and a genuine connection between both the conductor and ensemble as well as among the musicians themselves.

Just as Vivaldi's composition revolutionized the classical genre, today's ifeldof software development is experiencing a dramatic shift (left) in how teams approach security in order to adapt to the changing threat landscape.

Vulnerabilities *are* inherent to the nature in which software is constructed, so an application's composers must consider security at all stages of the SDLC — they must implement security practices into each movement of their craft to create not only a stable, secure product, but also to ensure a harmonious experience for the audience.

In the same way a solo violinist can introduce sounds that alter the performance of "Summer's" ending as it transitions to "Autumn," with its swift, rising tempo, an individual developer can introduce vulnerabilities into a feature through the composition of their code.

However, it is not only about a developer's focus on the ifne elements. Importance also lies in the wider team's commitment to innovative security practices and collaboration. And like that of a conductor, one who listens carefully to their ensemble's tempo shifts, to their pitch and concerted motion, those managing security-focused groups can be attuned to their team's collective needs as well as security vulnerabilities — both present ones and instances yet to be discovered.

With that awareness comes the invaluable ability to remain lfexible and agile in an ever-evolving threat landscape.

As the last few years have certainly shown us, through the exponential acceleration of digital transformation and (most especially) changes in how and where we work, security should be the crux of any application, system, and organization. So to you, I ask, "Are you ready?"

On behalf of the entire DZone family, I'm pleased to welcome you to our ifnalTrend Report of 2022, *Enterprise Application Security: Building Secure and Resilient Applications*. Our research explores how security has permeated all facets of software development, from both the technical perspective and within team and organizational cultures.

We also worked with several members from the DZone community who have contributed their expert insights on practices such as handling security incidents, applying zero-trust principles, and building security into the software supply chain.

Our hope is for the information within to help you set the stage and tune your instruments for ultimately delivering a secure product and smooth experience to customers.

Remember: The adventure is in the details, in the integrity of the production — of the performance itself.

Wishing you all the best,

Melissa Habit

ORIGINAL RESEARCH

Key Research Findings

An Analysis of Results from DZone's Application Security Survey

By G. Ryan Spain, Freelance Software Engineer, Former Engineer & Editor at DZone

In November 2022, DZone surveyed software developers, architects, and other IT professionals in order to understand the state of application security.

Major research targets were:

1. Perceived/observed application security risks
2. Development and testing techniques for securing applications
3. The role security plays for individuals within an organization as well as the SDLC as a whole

Methods: We created a survey and distributed it to a global audience of software professionals. Question formats included multiple choice, free response, and ranking. Survey links were distributed via email to an opt-in subscriber list, popups on DZone.com, the DZone Core Slack workspace, and various DZone social media channels. The survey was open from October 29–November 19, 2022 and recorded 535 complete responses.

Demographics: Due to the variations in respondent demographics between this year's and last year's Application Security survey, we've noted audience details below:

1. Only 6% of respondents described their primary role in their organization as "Developer/Engineer" (which was the largest category of respondents' roles last year at 35%). 25% of respondents said their primary role is "Technical Architect" (up from 20% in 2021), and 15% said their primary role is "Developer Team Lead" (down from 19% in 2021). Nearly one in five respondents (19%) said their primary role is "Site Reliability Engineer (SRE)" this year — a substantial increase from the 2% seen in last year's survey.
2. Far fewer respondents this year said they work on web applications/services (28%) than last year (77%). Most respondents for this year's survey said they work on enterprise business apps (62% in 2022 versus 45% in 2021); many also said that they work on boxed software, either with updates over the web (39% in 2022 versus 11% in 2021) or without web updates (25% in 2022 versus 9% in 2021).
3. Java was a much less popular language for respondents of this year's survey — only 17% of respondents said their company uses a Java ecosystem, compared to 75% last year. Client-side JavaScript — this year's most used language at 42% — was still lower than 2021's 62% response. C#, Node.js, Python, PHP, Scala, and TypeScript — all had signiifcantly lower results than last year. The only language with a statistically signiifcant increase this year was Ruby (15% in 2022 versus 11% in 2021).
4. Regarding responses on the *primary* language respondents use at work, last year, Java dominated the results at 52%, with Python (10%), C# (7%), and server-side JavaScript (6%) being the only other languages over 5%. This year, however, there is much greater variation in primary languages. Server-side JavaScript and C/C++ tied for the largest segments at 19% each, followed by client-side JavaScript (14%), Java (13%), C# (11%), Go (8%), and Kotlin (7%).
5. Generally, respondents this year had fewer years of experience as a software professional, with an average of 7.1 years of experience and a median of 5 years, compared to last year's average of 13.2 years and median of 12 years.
6. Respondents worked at generally smaller shops this year compared to last year, with more than 90% working at companies with less than 1,000 employees. Last year, 21% of respondents said they worked at companies between 1,000 and 10,000 employees, and 29% said they worked at companies with more than 10,000 employees.

In this report, we review some of our key research ifndings. Many secondary ifndings of interest are not included here.

Research Target One: Application Security Risks

Motivations:

1. Application security is deifned by risk; creating a secure application means ifrst understanding the potential threats that application faces. To that end, we wanted to determine which risks software professionals perceived as most problematic. We looked to OWASP's Top 10 Web Application Security Risks from 2021 to gauge how the vulnerabilities perceived by our respondents compared to the results of OWASP's security tests.
2. Security risks don't really present problems until an application is put into production. Code on the dev branch can still be ifxed before making it to main; code in the pipeline can have vulnerabilities patched before that code is deployed. While *in practice* security should generally be a consideration before the last possible moment, the consequences of poor security don't manifest until an application is made available to the user. In our survey, we wanted to ifnd out how often security issues made it through development and into the "real world."

COMPARISON TO OWASP TOP 10 WEB APPLICATION SECURITY RISKS

Every few years, the Open Web Application Security Project (OWASP) creates a top 10 list of application security risks, combining analysis of application data and a survey of industry professionals to note the most observed security issues in the web application space.

As described on their website: "The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications." We asked our respondents to rank each security risk in OWASP's Top 10 list by how problematic they found each risk:

In 2021, OWASP surveyed and identiifed the top security risks to web applications. Based on your experiences, please rank the following web application security risks in order of most problematic (top) to least problematic (bottom) for you and/or your organization:

Results:

Table 1

WEB APPLICATION SECURITY RISK RANKINGS

Risk	Rank	Score	n=
Broken access control	1	2,835	419
Identiifcation and authentication failures	2	2,621	442
Insecure software design	3	2,570	434
Vulnerable and outdated components	4	2,497	419
Injection	5	2,458	438
Logging and monitoring failures	6	2,419	427
Security misconifguration	7	2,295	421
Cryptographic failures	8	2,240	426
Software and data integrity failures	9	2,208	423
Server-Side Request Forgery (SSRF)	10	1,936	417

Observations:

1. Broken access control — number 1 on OWASP's Top 10 Web App Security Risks in 2021 — also presents the biggest issue for respondents.

   OWASP's number 1 spot in 2021 went to "broken access control," up from a middling position (#5) in 2017's report. Vulnerabilities under the umbrella of broken access control, according to the OWASP description, include: not following the "Principle of Least Privilege" or the "Deny by Default Principle"; accessing APIs with missing function-level access controls; improper privilege elevation; and manipulation of exposed metadata.

Our survey respondents, on average, also found broken access control to be the most problematic security risk of the 10, with the risk receiving the highest-ranking score of all the options available. According to OWASP's 2021 report, 24% of respondents ranked broken access control at number 1, beating OWASP's next-most number 1 rated risk, cryptographic failures, by 8%.

Perhaps relatedly, following the practice of the "principle of lowest possible privilege" was ranked second to lowest among 14 secure coding techniques in another question asked of our respondents, which we will address later in these research findings.

2. Identification/authentication failures and vulnerable/outdated components are ranked more problematic by our respondents than OWASP data indicates.

   Identification and authentication failures — then referred to as "broken authentication" — ranked number 2 in OWASP's previous top 10 report, from 2017. That category fell to the number 7 spot in 2021. According to OWASP: "This category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping." Vulnerable and outdated components — previously "using components with known vulnerabilities" — actually moved up in OWASP's rankings in 2021, reaching number 6 in the list from 2017's number 9 ranking.

   Both of these risks, however, were deemed to be more problematic by our survey respondents than seen in the OWASP list. Identification and authentication failures appeared at number 2 in our rankings, and vulnerable and outdated components came in at number 4. 41% of respondents ranked identification and authentication failures in their top three most problematic risks. And while almost no one ranked vulnerable and outdated components as the number 1 most problematic risk (only 2%), 40% ranked it second (11%), third (15%), or fourth (14%).

3. Cryptographic failures and injection are ranked less problematic by our respondents than OWASP data indicates.

   Cryptographic failures ranked number 2 overall in OWASP's 2021 top ten list, and injection — which was the number 1 spot of OWASP's 2017 list — ranked number 3 for OWASP in 2021. Cross-site scripting (XSS), formerly making up its own risk category in the 2017 OWASP list, was in 2021 added to the "injection" category.

   For DZone respondents, these categories both ranked as less problematic risks. Cryptographic failures, while being the second most common risk ranked in the number 1 spot (16% of respondents placed it highest on the list), overall ranked in eighth place. 26% of respondents ranked cryptographic failures last (16%) or next to last (10%). Injection came in fifth place in our survey rankings, and also saw an inverted bell curve in its results: 27% of respondents ranked injection first (15%) or second (12%), while 23% ranked it as last (15%) or next-to-last (8%).

VULNERABILITIES AFTER RELEASE

Like any other software bug, security issues are inevitable. Software is created by humans, and humans are imperfect, ergo software will be imperfect. What can be better controlled is where in the SDLC these security issues are discovered and rectified. Part of the SDLC — as well as the entire concept behind shift-left security — is to ensure that security issues don't make it to production code.

We wanted to see, however, how many security vulnerabilities were making it out of the development stage and actually being released to production. So we asked our respondents:

What percent of your releases introduce at least one security issue?

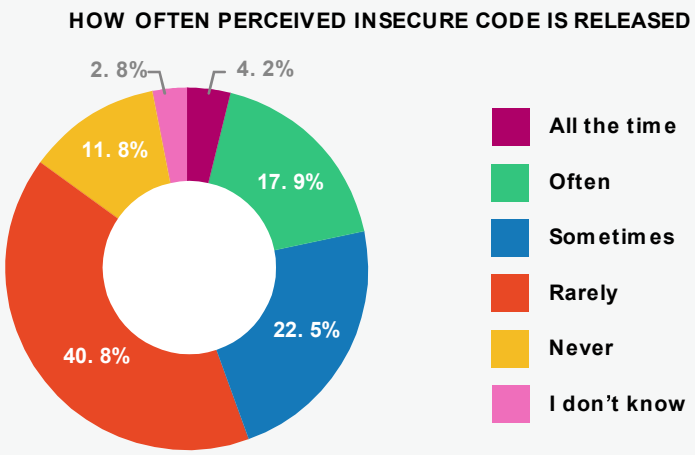
And:

How often do you release code that you are not confident is secure?

Results:

SEE FIGURE 1 ON NEXT PAGE

Figure 1



Observations:

1. Even by the estimates of those writing and deploying code, few releases occur without some potential security risk. While many respondents (41%) feel they rarely release code they are not conifdent is secure, only 12% feel they never do this. Almost half of respondents (45%) feel that they release code they are not conifdent is secure sometimes (23%), often (18%), or all the time (4%). 3% of respondents answered that they did not know.

   Though these conifdence levels do not speciifcally correlate to code vulnerabilities, they do help to paint the picture of application security's uncertain nature. The growing complexity of software design and development make it more and more dififcult to pick out security risks among thousands of objects, functions, API calls, services, libraries, etc. The ramiifcations of this dififculty can be seen when looking at how rarely software is released without introducing new security issues.

2. Only 2% of respondents (of 268 respondents answering, "What percent of your releases introduce at least one security issue?") said that none of their releases introduced any security issues, and only 12% of respondents estimated this to be less than 10%. On average, respondents claimed that 20% of releases introduced at least one security issue, with a median response of 13%.

   Respondents at the largest organizations (10,000+) had a lower estimate than most with regards to releases introducing security issues; those respondents, on average, said that 17% of releases introduced at least one security issue, with a median response of 5% (although the n value for this was quite low, with n = 17).

Research Target Two: The Role of AppSec in the SDLC and for the Developer

Motivations:

1. An application's development and deployment are made up of the efforts of individual software professionals; nothing gets coded, tested, built, released — or secured — without *someone* taking the responsibility for coding, testing, building, releasing, securing. But of course, these responsibilities vary by individual, and they are taken on for a variety of reasons, both internal and external to the individual (or even the organization).

   We wanted to see where respondents thought their organization put the onus of application security, as well as the factors impacting their own security decisions.

2. As mentioned previously, the concept of shift-left security is aimed at reducing security vulnerabilities that make it to production releases, seeking to ifnd and remedy issues earlier in the development process. The shift-left mentality makes sense when viewing the SDLC holistically or hypothetically, but in practice, it could put extra strain on developers and lead to dififcult decisions regarding priorities (If a developer must add security considerations to their own worklfow, what considerations must they cut back on? Does performance take a hit? Stability? Time to release?).

   We tried to ifnd out where in the SDLC organizations began implementing security, and whether respondents thought certain security considerations were occurring too early or too late in the SDLC.

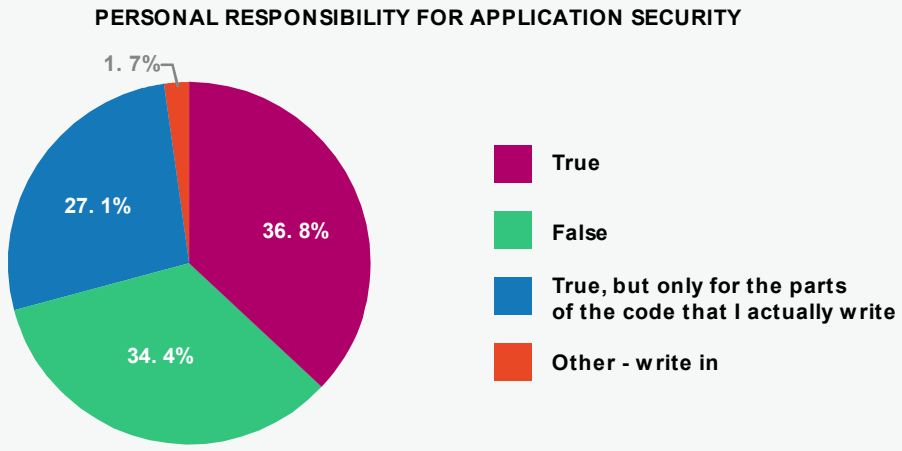
INDIVIDUAL RESPONSIBILITY AND FACTORS INFLUENCING SECURITY DECISIONS

Where individual responsibility lies in application development isn't always completely straightforward. Multiple people are often involved with the same code, and details regarding which individual has primary responsibility for an application's security may not be said outright. To find out whether respondents believed their organization placed responsibility for security on the respondents' shoulders, we asked:

In your employer's judgment, you are personally responsible for the security of any application you work on.

Results:

Figure 2



We also sought to find what forces influenced the respondents' own security decisions, asking:

How much impact do the following have on your application security decisions? Rank from greatest impact (top) to least impact (bottom).

Results are noted in Table 2.

Observations:

1. Respondents mostly believed that their employers consider them personally responsible for the security of the applications they work on, whether that responsibility lies with the respondent only for the code they write (27%) or for the application as a whole (37%).

   Still, many more respondents believe that their employers don't consider them responsible for the security of the applications they work on at all, with 34% of respondents answering "False" to the question compared to only 8% last year. This change may correlate somewhat with the 15% increase in respondents saying that their organization employs specialists in application security (up to 79% this year from 64% in 2021).

   In future research, we may be able to determine if this correlation holds, and can determine whether respondents' own opinions of their responsibility regarding application security aligns with their perception of their employers' position, or determine in what ways those opinions differ.

Table 2

FACTORS IMPACTING APPLICATION SECURITY DECISIONS

Factor	Rank	Score	n=
Regulatory requirements	1	2,853	409
An actual breach at your organization	2	2,598	418
Security awareness organizations (OWASP, SANS, etc.)	3	2,581	432
Customer requirements	4	2,519	412
Actual breaches at other organizations	5	2,418	429
Demands from investors	6	2,328	406
Demands from executives	7	2,241	413
Security makes our software more marketable	8	2,132	417
Users tell us about vulnerabilities in our software	9	1,986	430
Other	10	1,811	397

2. Like last year, regulatory requirements ranked as the most impactful factor influencing security decisions by a fairly wide margin. 18% of respondents rated regulatory requirements at number 1, while 50% of respondents ranked this option in their top three factors. This is perhaps a hopeful consistency, as it may mean that regulatory requirements are keeping up with threats as they emerge.

3. Our 2021 Application Security survey placed "an actual breach at your organization" at number 4 in the rankings list, and the option scored closely with software marketability and actual breaches at other organizations. This year, however, actual breaches at the respondents' organizations ranked number two, surpassing security awareness orgs like OWASP and customer requirements (numbers 2 and 3 in last year's rankings).

    This gives reason to believe that increasing complexity of security practices and development processes as a whole, or perhaps just an increasing number of security threats in general, have caused first-hand security breaches to become a wider experience for more developers.

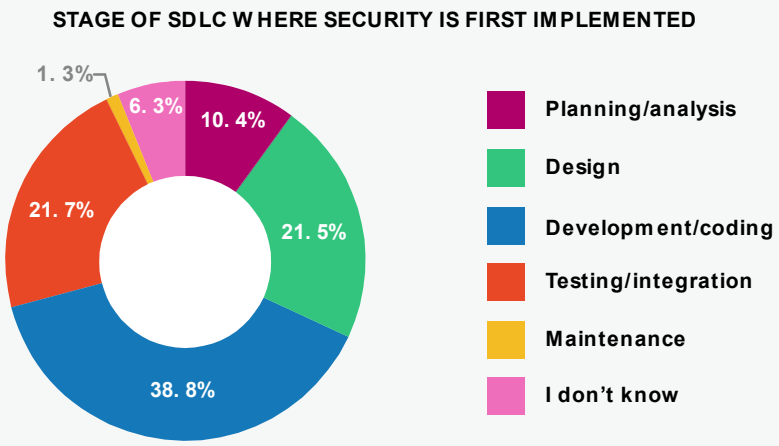
SECURITY'S PLACE IN THE SDLC

Leaving security as an afterthought in application development can be detrimental and cause security vulnerabilities to slip through to production code. Focusing on security too much too early may cause development time to increase, or it may require unnecessary resources to be added to a project or siphoned from somewhere else. To determine where in the SDLC organizations begin their application security considerations, we asked respondents:

At what stage in the SDLC does your organization first implement security?

Results:

Figure 3



In order to see how important respondents find security compared to other application development considerations, we asked:

In reality, how important are each of the following considerations? Rank from most important (top) to least important (bottom).

Results are detailed in Table 3.

Observations: We also wanted to determine software professionals' feelings on the shift-left security mindset. We asked our respondents' opinions on when they thought their organizations *should* explicitly consider security compared to where they typically see it now, as well as where they thought their organization *should* perform pen testing, fuzz testing, and static code analysis compared to when they actually perform these tests:

Table 3

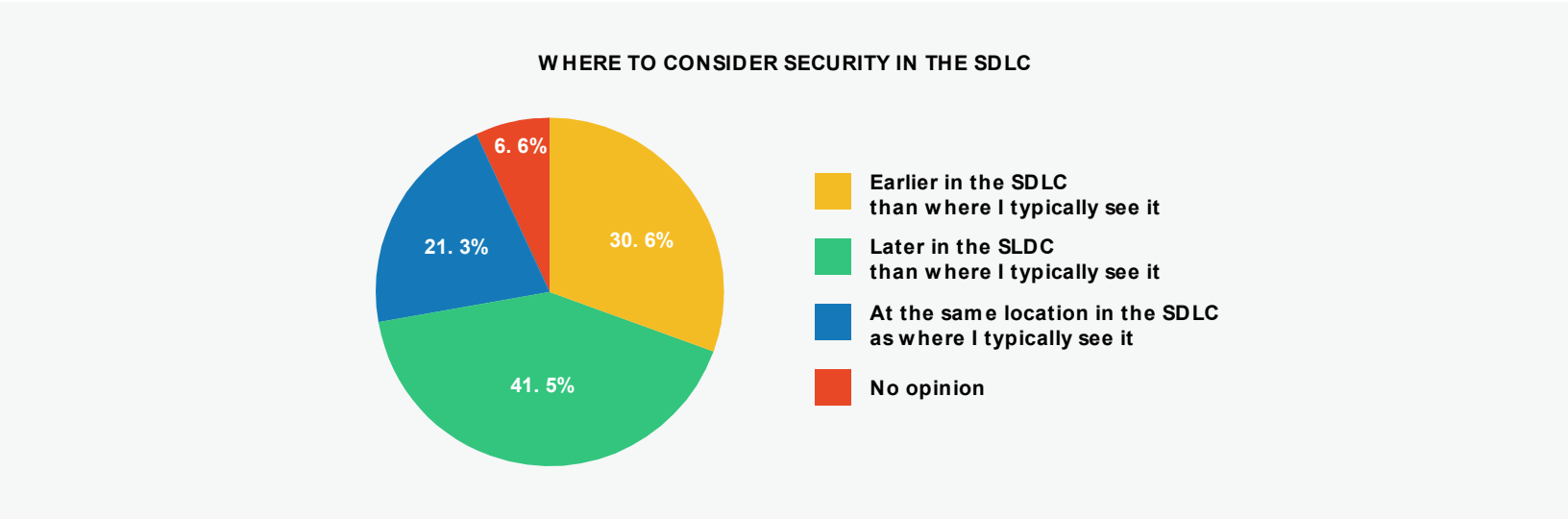
IMPORTANCE OF APP DEVELOPMENT CONSIDERATIONS

Item	Rank	Score	n=
Security	1	1,730	432
Maintainability	2	1,674	431
Performance	3	1,575	421
Technical aesthetics (of architecture, design, code itself)	4	1,448	420
Scalability	5	1,394	427
User experience	6	1,265	408

In your opinion, security should explicitly be considered:

Results:

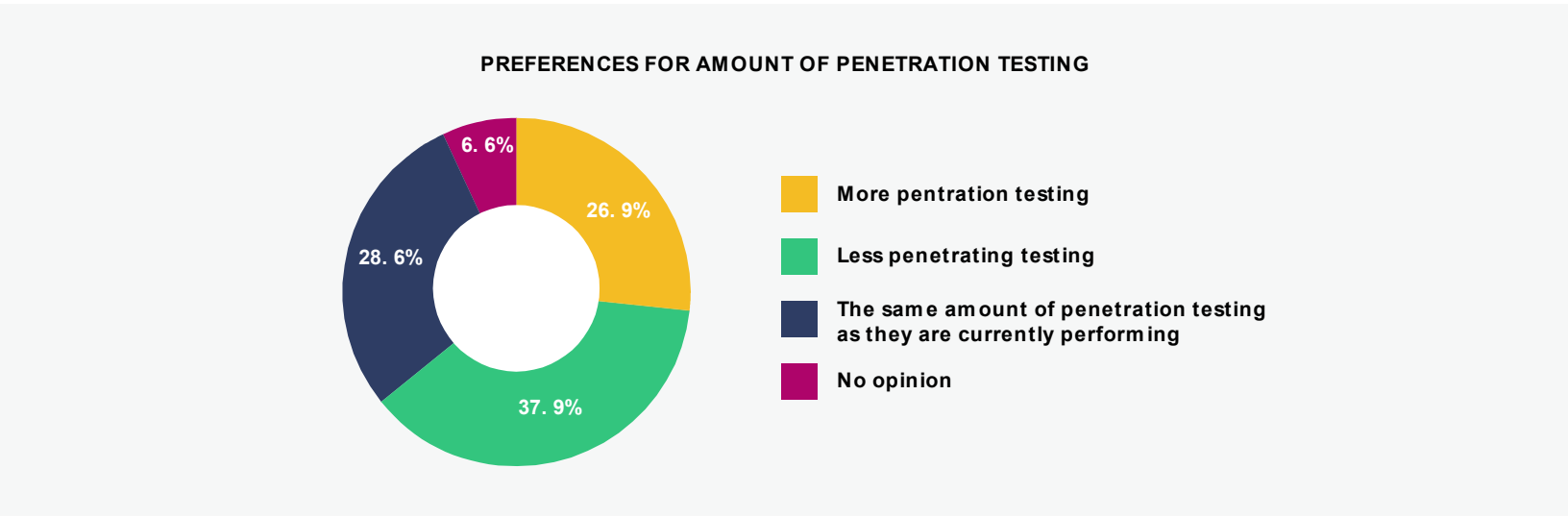
Figure 4



In your opinion, your organization should perform:

Results:

Figure 5



FIGURES 6 AND 7 CONTINUE ON NEXT PAGE

Figure 6

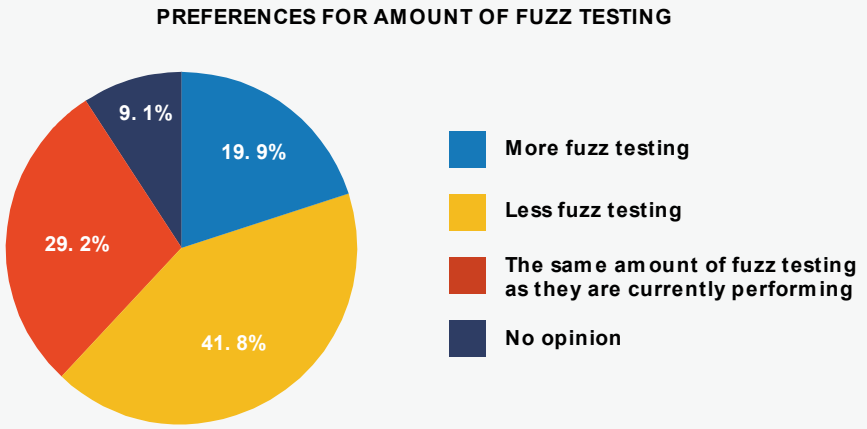
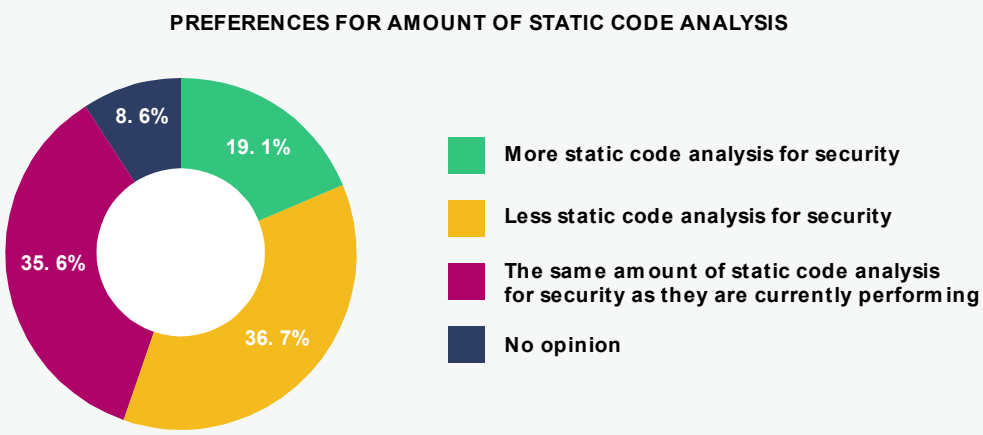


Figure 7



Observations:

1. Most respondents ifnd that their organization ifrst implements security in the development/coding stage or earlier, rather than leaving security considerations to testing, integration, or maintenance. 10% of respondents say their org ifrst implements security during the planning/analysis stage, 22% during the design stage, and 39% during the development/coding stage.

2. Security ranked highest when respondents were asked how important various considerations were in their software development over second-place maintainability and third-place performance. This deviates from last year's results, where respondents answered that security should be the most important consideration *in an ideal world*. But in reality, respondents ranked user experience and performance ifrst and second place, respectively, above third-place security. This newly placed importance on security could be correlated to the increase in ifrst-hand security breaches that organizations experienced, as mentioned earlier.

3. More than one-third of respondents believe that security should be explicitly considered later in the SDLC than they typically see it (42%) and believe that their organization should perform less penetration testing (38%), fuzz testing (42%), and static code analysis (37%).

    These results differ wildly from last year's survey, where 66% of respondents said that they believed security should be considered earlier in the SDLC than they typically see it (compared to 31% this year), and only 10% of respondents thought it should be considered later in the SDLC. And respondents last year mainly thought their organization should perform more penetration testing (62%), fuzz testing (52%), and static code analysis (58%), rather than less (6% for pen testing, 10% for fuzz testing, and 9% for static code analysis).

It is possible that an increased focus on security and stricter security measures throughout the SDLC have caused frustration for many software professionals, who must now adapt their workIfows around new and changing security-focused business requirements. Future research may be able to further elucidate the reasons behind this push back against the leftwards shift of security concerns in the SDLC.

Research Target Three: Security-Aimed Development and Testing Techniques

Motivations:

1. The ever-changing nature of the application development landscape — tools, techniques, practices, methodologies, etc. — brings change as well to the security risks those applications may encounter. Advancements in security necessitate new threats, which in turn require updated security techniques, and so on, ad inifnitum. We wanted to see what changes to security techniques and architectural patterns we could ifnd compared to last year's Application Security survey.
2. The integration of processes into DevOps pipelines is critical in much of modern enterprise development for maintaining quality control and facilitating smooth deployments. Automation of the processes necessary to release stable and safe software reduces the chance of human error and allows for testing at otherwise unreachable scales and speeds. We wanted to know how organizations were implementing security-related tools and tests into their DevOps pipeline.
3. How often to scan for security vulnerabilities varies from application to application, and "best practices" regarding scanning frequency depends on who you ask. How quickly vulnerabilities can be remedied once identiifed can also vary wildly, taking anywhere from hours to months. We tried to ifnd out how often organizations were performing these scans, and how quick their turnaround time was once a vulnerability was found.

CHANGES IN TECHNIQUES AND PATTERNS

As software development becomes, in general, more complex, and as threats evolve to try to outpace security precautions, techniques and patterns used to create more secure software must adapt. In order to understand how techniques and patterns have changed (and, in some instances, how software professionals' feelings about those techniques and patterns have changed), we compared answers between the 2021and 2022 survey responses for the following questions:

*How often have you seen **good** implementations of each of the following application security architectural patterns?*

Results:

Table 4

GOOD IMPLEMENTATIONS OF APPLICATION SECURITY ARCHITECTURAL PATTERNS: 2021vs. 2022

	Often		Sometimes		Rarely		Never		n=	
Pattern	2021	2022	2021	2022	2021	2022	2021	2022	2021	2022
Single access point	62.0%	17.1%	27.9%	60.0%	7.0%	16.9%	3.0%	6.0%	469	515
Check point	45.8%	21.9%	37.3%	39.9%	13.3%	32.2%	3.6%	6.0%	467	521
Security roles	57.3%	18.2%	32.5%	42.5%	8.1%	30.3%	2.1%	9.0%	468	522
Explicit sessions	29.4%	11.5%	32.0%	47.4%	27.9%	25.6%	10.7%	15.5%	459	523
Error messaging over hiding	30.8%	13.0%	32.5%	42.6%	24.4%	26.8%	12.2%	7.6%	467	524
Hiding over error messaging	45.5%	14.6%	31.7%	51.2%	15.3%	23.3%	7.4%	10.9%	470	514
Secure access layer	50.6%	17.8%	32.4%	41.8%	11.8%	31.2%	5.2%	9.2%	466	522

*How often have you seen **bad** implementations of each of the following application security architectural patterns?*

Results:

SEE TABLE 5 ON NEXT PAGE

Table 5

BAD IMPLEMENTATIONS OF APPLICATION SECURITY ARCHITECTURAL PATTERNS: 2021vs. 2022										
	Often		Sometimes		Rarely		Never		n=	
Pattern	2021	2022	2021	2022	2021	2022	2021	2022	2021	2022
Single access point	25.5%	15.3%	39.1%	51.5%	26.6%	26.0%	8.8%	7.2%	466	515
Check point	21.3%	15.6%	43.7%	35.5%	25.2%	41.3%	9.9%	7.6%	465	513
Security roles	24.6%	16.2%	38.4%	43.1%	27.2%	30.8%	9.9%	10.0%	464	520
Explicit sessions	22.7%	16.1%	38.1%	43.9%	26.8%	30.8%	12.4%	9.2%	467	522
Error messaging over hiding	27.0%	15.0%	38.7%	49.5%	24.8%	26.5%	9.6%	9.0%	460	521
Hiding over error messaging	22.0%	11.7%	40.9%	49.2%	26.5%	31.5%	10.6%	7.5%	464	520
Secure access layer	25.4%	12.1%	37.0%	42.3%	26.3%	33.7%	11.3%	11.9%	460	520

How often do you use the following secure coding techniques? Rank from most commonly used (top) to least commonly used (bottom).

Results:

Table 6

SECURE CODING TECHNIQUE RANKINGS: 2021			
Technique	Rank	Score	n=
Input validation	1	4,648	402
Secure coding standards	2	3,714	393
Data sanitization: input	3	3,626	384
Deliberate architecture and design sessions for security	4	3,532	372
Principle of lowest possible privilege	5	3,382	367
Whitelisting (permission explicit, denial default)	6	3,225	371
Static code analysis for security	7	3,040	382
Simple design to shrink attack surface	8	2,943	349
Data sanitization: output	9	2,599	345
Penetration testing	10	2,454	359
Policy of ignoring no compiler warnings	11	2,348	332
Threat modeling	12	2,093	320
Defense in depth	13	2,052	321
Fuzz testing	14	1,438	300

SEE TABLE 7 ON NEXT PAGE

Table 7

SECURE CODING TECHNIQUE RANKINGS: 2022			
Technique	**Rank**	**Score**	**n=**
Policy of ignoring no compiler warnings	1	3,933	423
Input validation	2	3,921	416
Static code analysis for security	3	3,810	425
Deliberate architecture and design sessions for security	4	3,509	424
Secure coding standards	5	3,336	419
Data sanitization: input	6	3,136	420
Penetration testing	7	3,020	413
Data sanitization: output	8	3,011	427
Whitelisting (permission explicit, denial default)	9	3,009	430
Defense in depth	10	2,984	417
Threat modeling	11	2,850	408
Simple design to shrink attack surface	12	2,805	414
Principle of lowest possible privilege	13	2,791	418
Fuzz testing	14	2,778	414

Observations:

1. In general, respondents of this year's survey were considerably less likely to answer "Often" or "Never," instead choosing "Sometimes" or "Rarely" when these answer choices were given.

   This could be indicative of an increased uncertainty around security practices, or a hesitance to subscribe to absolute beliefs as threats seemingly grow more and more possible in one's own application. It could be a side effect of the ifckle nature of securing software. Or it could just be a quirk in the samples between the two years, given differences in respondent demographics. Further research is necessary to make a sounder judgment on the reason(s) behind this difference.

   Note: Because of this variation, results in this section with Often/Sometimes/Rarely/Never answer choices will be compared aggregating "generally positive" results (often and sometimes) with "generally negative" results (rarely and never).

2. There seems to be much less conifdence in "good" implementations of security architectural patterns.

   Last year, 90% of respondents either sometimes or often saw good implementations of "Single access point" and "Security roles" patterns, compared to 77% and 61%, this year, respectively. "Secure access layer" had 23% fewer "often" or "sometimes" answers this year, and "Check point" dropped 21% this year. The only choice that did not fall a signiifcant amount since last year was "Explicit session," which only fell 2%. Still, a majority of respondents said they found good implementations of all given patterns either sometimes or often rather than rarely or never.

   This year's results for "bad" implementations of security architectural patterns were much more in line with last year's. "Single access point," "Explicit sessions," "Error messaging over hiding," and "Hiding over error messaging" saw less than a 3% change in respondents saying they had seen bad implementations of these patterns either often or sometimes, within our margin of error for this sample. "Security roles" had 4% fewer respondents observing bad uses of this pattern often or sometimes, while "Secure access layer" had 8% fewer and "Check point" 14% fewer, so the few patterns that did see year-over-year change saw overall fewer bad implementations than last year.

3. "Policy of ignoring no compiler warnings" jumped from an 11th place ranking in 2021to ifrst place in secure coding techniques in 2022, just beating out last year's number-one spot, "Input validation." Also with rankings several spots higher this year, "Static code analysis" climbed from seventh to third place; "Penetration testing" went from 10th to seventh place; and "Defense in depth" moved from 13th to 10th place.

4. As mentioned, "Input validation" dropped from the top of the list to number 2 since last year, though the ranking scores between ifrst and second place this year were within about 0.3%. While 50% of respondents this year ranked input validation in their top three (as opposed to ignoring no compiler warnings at 41%), 12% of respondents put input validation last, lowering its score.

5. "Principle of lowest possible privilege" had the biggest drop in rank from last year, falling from iffth place in 2021 to 13th place — next to last — in 2022. Other techniques that fell signiifcantly in rank were: "Simple design to shrink attack surface," moving from 8th to 12th place; "Whitelisting," going from sixth to ninth place; "Secure coding standards," dropping from second to iffth place; and "Data sanitization," going from third to sixth place.

6. The reasons for these changes in technique usage could be many. Some techniques may have proven more effective than others and increased in popularity. It is possible that some of these techniques have become more automated and are taking less collective space in software professionals' minds. Or considering the differences in demographics between surveys, certain techniques may be better suited to certain industries, or they are implemented more or less obviously in certain languages. Future research may help to determine what is impacting the adoption of these techniques.

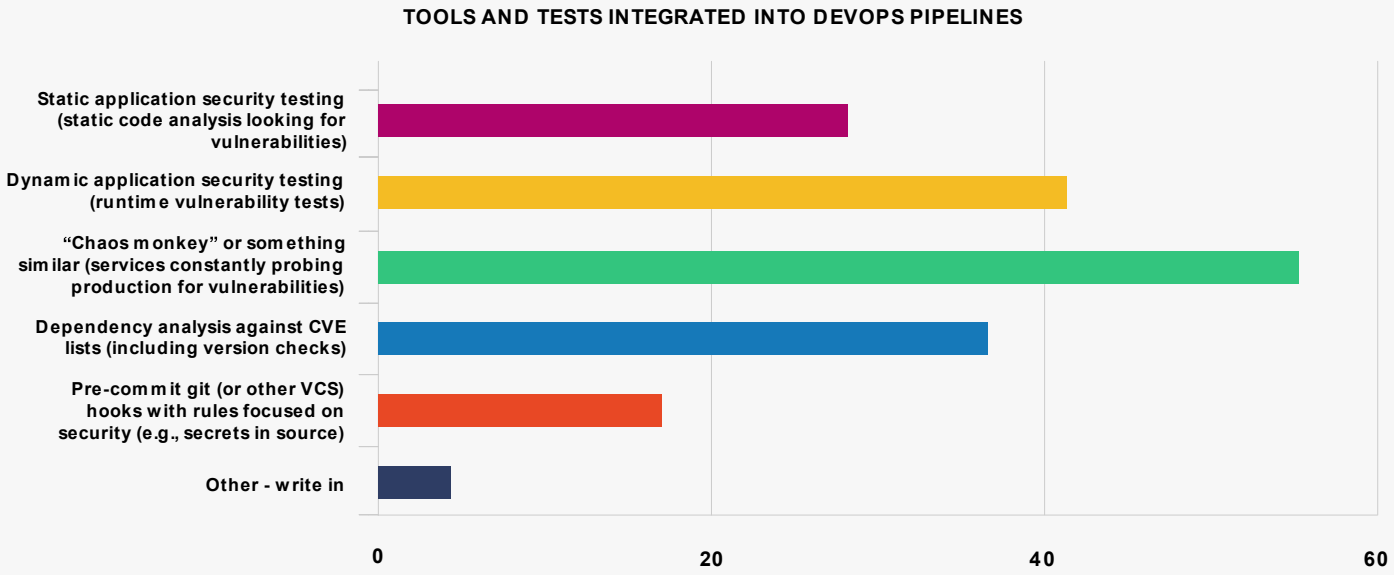
TOOLS AND TESTS IN THE DEVOPS PIPELINE

To ifnd out how organizations are utilizing their DevOps pipeline for application security, we asked:

Which of the following security tools/tests are integrated into your DevOps pipeline?

Results:

Figure 8



Observations:

1. Production-probing services (such as "Chaos monkey") are currently the most popular tools integrated into DevOps pipelines — more than half of respondents (55%) said their pipelines have "Chaos monkey" or something similar. In 2021, only 27% of respondents said their DevOps pipelines had this type of tool. The nature of chaos engineering in general makes it a prime candidate for DevOps integration; these tools work by running constantly and randomly on production servers, and the computer automation required to make this work goes hand in hand with DevOps pipeline automation.

   Dynamic application security (44%) and dependency analysis against CVE lists (37%) were the next most popular tools/tests in DevOps pipeline integrations.

2. Besides Chaos-monkey-like probes, all options received a lower percentage of responses this year than last year. In particular, static application security testing, last year's most popular response at 75%, had only 27% this year.

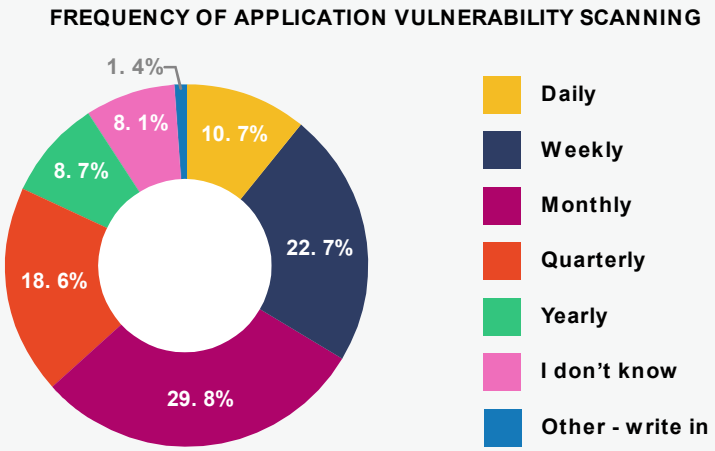
FREQUENCY OF SCANNING AND TURNAROUND TIME FOR ADDRESSING CVEs

Scanning applications for vulnerabilities is vital to keeping software secure. New threats are constantly being identiifed, and reliance on third-party libraries and services means that breaches don't have to come from your application's code. To see how often organizations are scanning their applications, we asked:

How often is your organization scanning applications to detect and identify vulnerabilities?

Results:

Figure 9

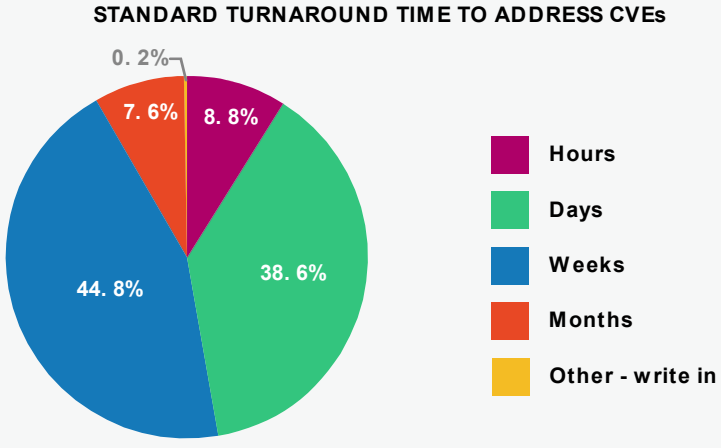


We also wanted to see what kind of turnaround time organizations were achieving once common vulnerabilities and exposures (CVEs) were identiifed. We asked respondents:

What was the standard turnaround time for addressing CVEs?

Results:

Figure 10



Observations:

1. Most respondents said their organization scans for vulnerabilities at least once a month. 30% said their scans occur monthly, 23% said they occur weekly, and 11% said they occur daily. Only 19% said that these scans occur once per quarter, and just 9% said they occur once a year.

2. Most organizations are able to address CVEs in either days (39%) or weeks (45%). Very few respondents said that their standard turnaround for addressing CVEs took only hours (9%) or took months (8%). Given the severity of many security vulnerabilities, these turnaround times overall seem slower than one might hope or expect.

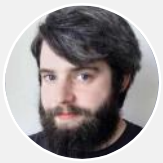
   However, this question does not take severity into account, which could have signiifcant impact on how long it takes to address a CVE (one would imagine that a critical vulnerability, or one known to actively be exploited, would take less time to address than a non-critical issue). Still, even vulnerabilities seen as non-critical could be one exploit away from becoming a critical one.

Future Research

There's still much more to analyze from our survey results and plenty of opportunity to reifne and expand our Application Security survey in the coming years. Some topics we didn't get to in this report, but were incorporated in our survey, include:

- Writing application code to verify message integrity
- Securing legacy code vs. new code
- Reliance on third parties to maintain secure code
- Service level agreements to address common vulnerabilities and exposures
- Techniques for detecting intrusion attempts
- Using custom HTTP headers for sender veriifcation

Please contact publications@dzone.com if you would like to discuss any of our ifndings or supplementary data.



G. Ryan Spain, Freelance Software Engineer, Former Engineer & Editor at DZone

@grspain on DZone | @grspain on LinkedIn, GitHub, and GitLab | gryanspain.com

G. Ryan Spain lives on a beautiful two-acre farm in McCalla, Alabama with his lovely wife and adorable dog. He is a polyglot software engineer with an MFA in poetry; a die-hard Emacs fan and Linux user; a lover of The Legend of Zelda; a journeyman data scientist; and a home cooking enthusiast. When he isn't programming, he can often be found watching reruns of *Star Trek: The Next Generation* with a glass of red wine or a cold beer.

The Best Cloud Security is Now Even Smarter!

Today's cloud environment needs more context in order to provide smarter security—**FAST**.

From code to cloud, Check Point CloudGuard delivers unified cloud native security across your applications, workloads, and network to manage risk, maintain posture and prevent threats.

CloudGuard
CNAPP

Effective Risk Management Engine

- Network Security
- Code Security
- WAAP
- CSPM
- CWPP
- CIEM

For more information on CloudGuard, visit:
www.checkpoint.com/cloudguard

CHECK POINT™
YOU DESERVE THE BEST SECURITY

PARTNER OPINION

Cloud Multi-Tenants: 5 New Security Headaches in Store for DevOps

By Dotan Nahum, Head of Developer-First Security at Check Point

As cloud multi-tenancy increases, so do the security concerns for DevOps teams. To address these new security headaches, DevOps teams must proactively implement effective security measures and regularly review and update them to protect sensitive data.

Security Headache #1: Secure Access Controls

The complexity of managing access controls can create additional security overhead, increasing the risk of security incidents. Here are a few ways to increase access controls for cloud multi-tenancy:

- ☐ Implement strict authentication policies using role-based access control so that only authorized users can access sensitive data and resources.
- ☐ Implement network segmentation and access controls to isolate sensitive data and resources, preventing unauthorized access between different tenants and networks and limiting lateral movement within the network.
- ☐ Monitor and audit access to ensure that controls are being properly enforced and unauthorized access is detected and prevented. By using scanning tools, you can automate this process and ensure that the system remains secure.

Security Headache #2: Data Isolation

Data isolation is a problem for cloud multi-tenancy because it is dififcult to make sure the data belonging to one tenant is not accessible by other tenants sharing the same physical infrastructure. Here's how to increase data isolation:

- ☐ Use virtual machines and containers to create isolated environments for each tenant and ensure it cannot be accessed without proper authorization.
- ☐ Use encryption to protect sensitive data in transit and at rest and to prevent unauthorized access to data, even if it is intercepted or stolen.

Security Headache #3: Data Protection

A lack of data protection can lead to data breaches and other security incidents, which can have serious legal and reputational consequences, as well as violate data protection laws and regulations for both the cloud provider and the affected tenants. So how can you increase data protection in a cloud multi-tenancy environment?

- ☐ Conduct regular security assessments and penetration tests to identify and address cloud infrastructure and application vulnerabilities.
- ☐ Develop and implement robust disaster recovery and business continuity plans to ensure your business can continue operating in the event of a security incident and recover quickly.

Security Headache #4: Audit Logs

Not every company has the resources and expertise to effectively manage and analyze their audit logs, but it can be achieved by following these steps:

- ☐ Implement a centralized logging system to collect and store audit logs from all systems and applications.
- ☐ Use automated tools to monitor and analyze audit logs in real time, identifying potential security issues, such as unauthorized access or suspicious activity, and alerting security teams to take action.
- ☐ Regularly review and analyze audit logs to identify trends and patterns that may indicate security issues.

Security Headache #5: Resource Utilization

Resource utilization can impact shared infrastructure performance and reliability, creating vulnerabilities that attackers can exploit. Here's how to manage resource utilization in a cloud multi-tenancy environment:

- ☐ Use monitoring and management tools to track resource utilization in real time and identify potential issues; this enables corrective action to be taken.
- ☐ Implement resource allocation and policies, such as setting limits on the number of resources that each tenant can use and ensuring resources are shared fairly between tenants.
- ☐ Use automation and orchestration tools to manage and optimize resource allocation and utilization so tenants are able to access the resources they need when they need them.

You can get a checklist here to help your team ensure that your vulnerable spots are secured.

CONTRIBUTOR INSIGHTS

DevSecOps Easter Eggs

Unearth Your Security Wins

By Julie Tsai, Cybersecurity Leader, Board Member, & Investor/Advisor

Security can often feel like a Sisyphean endeavor. Day after day, we roll what feels like a growing pile of rocks up the hill, building resilience from threats, internal weaknesses, resourcing challenges, hostile politics, hubris, laziness, inertia. Sifting through potentially overwhelming issues — as well as *too many tools* — is a steady discipline cybersecurity professionals must master. Not to mention the accumulated fatigue that can cause you to gloss over the details and daily doings.

But some of the deepest, most important wins can happen when you cut through the noise, get down to what matters, and consider daily routines with more attention and purpose. It's not a new line item on the budget or a new unicorn hire. Instead, it requires focus and a relatively small investment of time to reconsider and refactor regular routines, and this can pay big dividends. Often, the most critical security holes are opened during regular coding, building, and deploying.

Perfection is impossible, but putting care and attention into the work *is* possible. That's where we should aim. Being aware of the blind spots in an organization (and people) can go a long way toward defending against the unknowns. But where to start?

What Is DevSecOps and Why Does It Matter

DevSecOps is an evolution from DevOps — the idea of integrating core principles of development and operations teams into each other's practices and toolsets. In that process of incorporating a developer mindset into operations, as well as an operations mindset into development, security must not be forgotten. The joke: "DevOps is when you give the root keys to the developer" is dark humor for security professionals who have seen many interpretations of DevOps agility where security and resilience are forgotten or an afterthought. Truly excellent DevOps means that security is *integrated* into DevOps from the beginning, not deprioritized in order to conduct more insecure, unstable projects faster. The art is accomplishing this without sacriifcing agility and lfexibility.

Because there are numerous ways companies can implement DevOps, it often feels like taking a Rorschach test — where the type of implementation is an answer that illustrates how the company views the importance of operations fundamentals relative to development. For example, of the many deifnitions for DevOps, DevSecOps can be realized as an extension of the core deifnitions of DevOps.

What DevSecOps *Isn't*

Just as importantly, what *isn't* DevSecOps? DevSecOps is not just deployment (or a secure deployment). A lot goes into producing secure code before and after it gets to production, meaning the code becomes accessible to the customer. Ask yourself:

1. Is the build itself — code and conifgurations — secure?
2. What new access is needed to services and have those been secured?
3. What pathways will critical data take, and what can access that data?

Likewise, a lot happens *after* code goes to production. During this stage, ask your teams:

1. What changes as applications run and build up load and stress?
2. Does that change the security of the application?
3. What attack vectors are most common?
4. What does the code depend on to run reliably and securely?

Deployment is a critical gate — one of the most important ones — but it's merely one step in the lifecycle of a project. The whole journey must be looked at, including what is picked up along the way in terms of data and access, and what is potentially lost.

DevSecOps and Its Easter Eggs

DevSecOps is a value stream of integrating security needs into the tools that engineers use to build and deploy. Within that, there is a wealth of opportunities (or landmines) to do things securely and efifciently. Or its reverse. As it is a fundamental virtue of technology for the tech to be independent of the policy, any tool can be used for good or for ill. It all depends on the practitioner and the care with which it is handled.

In the following section, we will hunt the ifve main DevSecOps Easter eggs. Easter eggs in this context refer to a hidden joke or treasure often found in video games which rewards the user as an insider. The reward for the Easter egg hunter is foiling potential security attackers.

1. SECRETS, PASSWORDS, AND CREDENTIALS: HIDDEN IN PLAIN SIGHT

One doesn't need to look far for major public breaches to see why this is important, e.g., hardcoded passwords and tokens in Travis CI or an Uber developer's private GitHub repo containing credentials to critical company apps and data that led to 57 million user and driver account compromises — and major regulatory and legal impacts. So what Easter eggs should we look for when it comes to passwords and credentials? You don't want credentials hiding in the following places, so take this opportunity to ifx major vulnerabilities.

1. **Hardcoded passwords and tokens –** these must go in a secured password manager or use a standard, hardened authentication/authorization system
2. **Files on systems –** this consists of ownerships and application access
3. **Shared credentials and keys**
4. **Credentials and keys stored in source control**
5. **Credentials loaded in app or memory –** this includes credentials that may be deifned in environmental variables, as well as those in applications that are left running indeifnitely and broadly accessible
6. **Unchanged default credentials –** this can be vendor or institutional
7. **Weak credentials –** this is vendor as well as core system credentials

2. HOW WIDE IS YOUR CIRCLE OF TRUST?

For critical information or operations like deployments or permissions changes, *do you know everybody who can touch or access production to deploy or change key permissions/creds*? *Do you know every app*? (See previous section on passwords and credentials stored in running instances or environments.) Tools developed to create no-friction deployment have made it hard to maintain cross-department visibility over time.

We have to get rid of that old canard "security by obscurity," especially with regard to this last point. Obscurity is not security — vulnerabilities have lifetime exposure: the attacker who can lie in wait has the luxury of time for reconnaissance. The attacker who knows your weaknesses and habits very well has their pick of what to exploit and when. And it is foolish hubris to assume that outsiders would have a harder time understanding or exploiting your systems than insiders. Outsiders often have the advantage of fresh eyes and sharpened toolkits to exploit common mistakes, even those made by smart people. Don't make it easy for an enemy to gain insight on you.

3. YOUR GOLD REPO SOURCE(S) OF TRUTH

Your source control repository should be considered your crown jewels: Not only is your intellectual property there, but your coding patterns, architectural insights, and occasionally (unfortunately) critical data resides there as well. Some key questions to ask regarding how well your repositories are being guarded and maintained include:

1. How are you securing your software?
2. Do you have known vetted repos?
3. Do you trust these?
4. How are they maintained?
5. What software is allowed to be on there?
6. Do you have a Bill of Materials?
7. Can you guarantee its secure provenance — its history and pathway?

It is a behemoth to be considered, but you get a lot of bang for the buck in securing code repositories and the pathways in and out of there. This gives not only a good DevOps architectural view but also provides key insights on your main leverage point for change, security, deployments, *and* reliability.

4. MEMORIES: WHAT DOES YOUR APP (AND SYSTEM) REMEMBER?

Credentials and other sensitive information — credit card information, private personally identiifable information (PII), customer data, sensitive IP intellectual property (and sometimes literal IP addresses) — are often stored in app or instance memory. How well is this being handled and secured? Ask yourself:

1. When are references cleaned out, access controlled, garbage collected, or restarted?
2. What controls information from being dumped from memory or improperly accessed?

This is a more subtle yet dangerous point precisely because of the more tenuous state of logging or easily accessible forensics for data and operations in memory. Attackers that gain access to the memory of applications and systems can also gain visibility into an arbitrary volume and range of data that can be hard to pinpoint. Limiting or managing disclosures for this upper bound of information could get ugly. Help nip it in the bud by making sure sensitive data is encrypted wherever possible; decryption is handled precisely with layered defenses and strongly defended keys, and access controls and forensic logs are well-instrumented on sensitive systems.

5. STORMY WEATHER: ARE YOUR CLOUD HATCHES BATTENED DOWN?

Everything's going to "the cloud." Great. But have you heard that quip saying the cloud is just a name for someone else's computer? There's truth to that. It's basically a leased or rented stack that (hopefully) has been equipped with more lfexible scaling tools than the one had originally. What does that mean for companies? Many teams and organizations gain an illusory ease in spinning up and scaling compute power. But do they understand what they're spinning up, how they're constructed, and where they are vulnerable?

There are some key questions to consider and answer as you secure your cloud stack (and orgs):

1. Who or what gets to deploy stuff — code, builds, changes, or new instances — into your cloud network?
2. Have the application-to-application access patterns (risks) been vetted?
3. Have instances been secured and hardened? (e.g., unnecessary services shut down, programs stripped off, network rules vetted, logging tuned up, default creds changed, access systems integrated, etc.)
4. Are all administrators of the systems *truly* administrators?

The evolution of DevSecOps lets engineers stretch their wings to expand into new competency areas across all three pillars: development, security, and operations. Developers build out full-stack capability by constructing applications from start to ifnish and understanding the full lifecycle; every operations or system administrator must gain programming proifciency with at least one language. And any of those who neglect security will often learn the hard way how to master it.

Conclusion: What Does This All Mean?

The connecting theme is *trust*, but not blind nor even zero trust — instead, earned trust. Defense-in-depth is a team sport and a full-stack discipline, all the way from hardware to humans. Take all of that into account when setting your org's priorities. For example, ask:

- How do you know to trust *where* your toolchain is secure?
- How do you know to trust *what* has been hardened against a vulnerability?
- How do you know to trust *who* is making changes and commits to your systems?

Asking good questions can be as important —or more — as coming up with the answers. It gives the opposing party the room to develop their own pathways and muscle memory to a solution, and it can open up new lines of inquiry that previously had been hidebound in its own assumptions. Asking good questions allows room for new solutions to evolve that beneift — and even improve — the original solutions.

Julie Tsai, Cybersecurity Leader, Board Member, & Investor/Advisor
@lfying_bat on DZone | @julietsai on LinkedIn | @446688 on Twitter | julietsai.net

Julie is a six-time CISO/Head of InfoSec and DevOps(Sec) specialist with 25+ years in international and Silicon Valley tech. She was hands on for 15 years, before transitioning to management. She's been chief/senior security leader with Roblox, Walmart eCommerce, Adteractive, and Box. A Stanford graduate, she is on the boards of the Bay Area CSO Council and other non-proift organizations.

CONTRIBUTOR INSIGHTS

Software Supply Chain Security Checklist

5 Keys to Keeping Software Secure From Creation to Delivery

By Justin Albano, Software Engineer at IBM

Software security is a critical part of any product, but it can be a second-class consideration for many projects when time and budget pressures are applied. It is not enough to deliver working software; our deliveries must also be trustworthy and secured against known vulnerabilities. For many applications, the security of thousands or even millions of Personal Information (PI) records — and even entire company networks — rests upon the security of our applications.

Since the stakes are so high, we must perform our due diligence when securing our products against vulnerabilities and earn the trust of our customers and users.

In general, vulnerabilities can come from our application code and dependencies (the *software supply chain*). We must be deliberate about the dependencies we introduce into our product and ensure they do not compromise our application.

5-Step Checklist for Securing Your Software Supply Chain

The checklist below details ifve simple but essential steps that we can take to ensure that our software supply chain is secure and that we are delivering the most trustworthy software possible to our customers and users.

❑ 1. BE PROACTIVE

As professionals, we are responsible for securing our products. When we release a product, we put our names and seals of approval on the product. We are fallible, but when we release our product to customers with vulnerabilities and are not diligent in ifxing them, that relfects poorly upon our company and us.

Even when vulnerabilities come from the dependencies we include in our product, it is still our responsibility to patch our application and remove the vulnerabilities. We have decided to use those dependencies, and we must, therefore, assume the responsibility of adequately vetting and upgrading those dependencies, especially when known vulnerabilities arise. Ultimately, if our product compromises a user's or customer's security, the fault rests with us, and we must do everything within our purview to protect our users and customers.

❑ 2. KNOW EVERY DEPENDENCY

Any code we include in our application — even transitive dependencies pulled in by our dependencies — can compromise our application. Therefore, it is our responsibility to enumerate our application's dependencies. This enumeration is sometimes called a *Software Bill of Materials* (BOM) and is analogous to a BOM in more traditional sectors, such as manufacturing.

We can ifnd a list of all dependencies (including transitive dependencies) in our application using the command-line interface for our package manager. For example:

- Maven – `mvn dependency:tree`
- Gradle – `gradle -q dependencies`
- NPM – `npm list --all`

We must keep our BOM up to date and in sync with the dependencies included in our product. One of the simplest ways to ensure this consistency is to generate our BOM as a step in our automated build pipeline. Each time we build our application, our build system produces a new BOM. As we add, change, or remove dependencies, the BOM generated for each build will be updated accordingly.

❑ 3. TRACK THE LATEST VULNERABILITIES

Once we know what our product depends on, we must track the security state of those dependencies. Keeping up to date with security bulletins and the Common Vulnerabilities and Exposures (CVE) system can be daunting, but there are resources we can use to help us.

The following are some of the most common databases that contain up-to-date lists of existing software vulnerabilities:

- GitHub Advisory Database
- US National Vulnerabilities Database
- MITRE CVE Database

❑ 4. AUTOMATE THE PROCESS

Even for the most trivial project, keeping tabs on our dependencies and cross-referencing them with security bulletins can be overwhelming. We should automate these security steps as much as possible, and our build pipeline should run them each time we build our product. Whenever possible, we should also leverage the tools already available to us.

For example, if we host our application on GitHub, we can use Dependabot, and if we host our application on GitLab, we can use dependency scanning.

❑ 5. AUDIT DEPENDENCIES

Every dependency in our product brings with it a set of vulnerabilities. Even the most ubiquitous dependencies, like log4j, can be compromised (see Log4Shell). Therefore, we should always have a minimalist approach to our dependencies. If a dependency is unused, we should remove it immediately, and if the security risk posed by a dependency outweighs its reward, we must ifnd an alternative. Finding unused dependencies and generating warnings for such dependencies can often be performed automatically (e.g., `mvn dependency:analyze` for Maven) and should be added as a step in our build pipelines.

Conclusion

We are responsible for the security of the products we deliver, even when vulnerabilities come from dependencies in our supply chain. Our customers' and users' trust in our products rests upon our diligence in ifnding and patching the vulnerabilities within our code and supply chain. Maintaining a professional mindset, understanding our exposure, tracking the latest vulnerabilities, automating our processes, and auditing our products go a long way in ensuring that our products meet the highest security standards and protect the data and systems of our users.

Justin Albano, Software Engineer at IBM

@albanoj2 on DZone | @justin-albano on LinkedIn

Justin Albano is a Software Engineer at IBM responsible for building software-storage and backup/recovery solutions for some of the largest worldwide companies, focusing on Spring-based REST API and MongoDB development. When not working or writing, he can be found practicing Brazilian Jiu-Jitsu, playing or watching hockey, drawing, or reading.

CONTRIBUTOR INSIGHTS

Improve Microservices Security by Applying Zero-Trust Principles

By Apostolos Giannakidis, Principal Product Security Engineer at Microsoft

According to a 2020 Gartner report, it is estimated that by 2023, 75 percent of cybersecurity incidents will result from inadequate management of identities and excessive privileges. To a large extent, this is attributable to the increased number of identities used by modern cloud infrastructures. Applications run as microservices in fully virtualized environments that consist of dynamically orchestrated clusters of multiple containers in the cloud.

The security requirements in such environments are signiifcantly different compared to monolithic applications running on premises. First, the concept of the perimeter does not exist in the cloud. Second, organizations are now handling thousands of dynamically created workloads and identities. Applying traditional IAM tools to manage the dynamic nature of these identities is not adequate. Using static, long-lived, and often excessive access permissions enables attackers to perform lateral movement.

To address these issues, a security model is needed that better satisifes today's application security and identity requirements. Zero-trust security is a proactive security model that uses continuous veriifcation and adaptive security controls to protect endpoints and access to applications as well as the data that lfowsbetween them. Zero trust replaces the outdated assumption that everything running inside an organization's network can be implicitly trusted. This security model has proven to minimize the attack surface, offer threat protection against internal and external attackers, reduce the lateral movement of attackers, increase operational efifciency, and help support continuous compliance with regulations such as PCI-DSS and the White House's 2021Cybersecurity Executive Order.

Since its inception, zero trust has evolved and expanded, touching almost every corner of the enterprise. This article will provide an overview of how the zero-trust principles can be applied in a microservices environment and what security controls should be implemented on the back end.

Zero-Trust Principles

Zero trust is primarily based on the concepts of "never trust, always verify" and "assume everything is hostile by default." It is driven by three core principles: assume breach, verify explicitly, and the principle of least privilege.

ASSUME BREACH

Always assume that cyber attacks will happen, the security controls have been compromised, and the network has been inifltrated. This requires using redundant and layered security controls, constant monitoring, and collection of telemetry to detect anomalies and respond in real time.

VERIFY EXPLICITLY

No network trafifc, component, action, or user is inherently trusted within a zero-trust security model, regardless of location, source, or identity. Trust only to the extent that you verify the identity, authenticity, permissions, data classiifcation, etc.

PRINCIPLE OF LEAST PRIVILEGE

Always grant the least number of privileges. Only give access for the time that it is needed and remove access when it is not needed anymore. Least privilege access is essential to reduce the attack surface, limit the "blast radius," and minimize an attacker's opportunity to move laterally within an environment in case of compromise.

Zero-Trust Security in a Microservices Environment

When a microservice is compromised, it may maliciously inlfuence other services. By applying the principles of zero trust to a microservices environment, the trust between services, components, and networks is eliminated or minimized.

IDENTITY AND ACCESS MANAGEMENT

Identity and access management is the backbone of zero trust, which requires strong authentication and authorization of end-user identities, services, functions, workloads, and devices. To enable authentication and authorization, we must ifrst ensure that each workload is automatically assigned a cryptographically secure identity that is validated on every request. Importantly, ensure that there is an automated mechanism to reliably distribute, revoke in case of compromise, and frequently rotate the services' certiifcates and secrets. Use a cloud-neutral identity for workloads, such as SPIFFE for authentication and OPA for uniifed authorization across the stack.

SECURE SERVICE-TO-SERVICE COMMUNICATIONS

In zero trust, it is fundamental to treat the network as adversarial. Thus, all communication between services, APIs, and storage layers must be encrypted. The standard way of protecting data in transit is to use HTTPS and strict mTLS everywhere. Similarly, a strong authentication mechanism should be enforced across all microservices. It must be understood that not every service that can be authenticated should be authorized. Authorization must be done based on the authentication context and on access control policies, and it should be performed at the edge of each microservice — not at the network edge.

To achieve this, use a service mesh, like Istio or Linkerd, for:

- Automatic certiifcate management
- Trafifc interception
- Secure service-to-service communication without application code changes
- Micro-segmentation (via authorization policies)

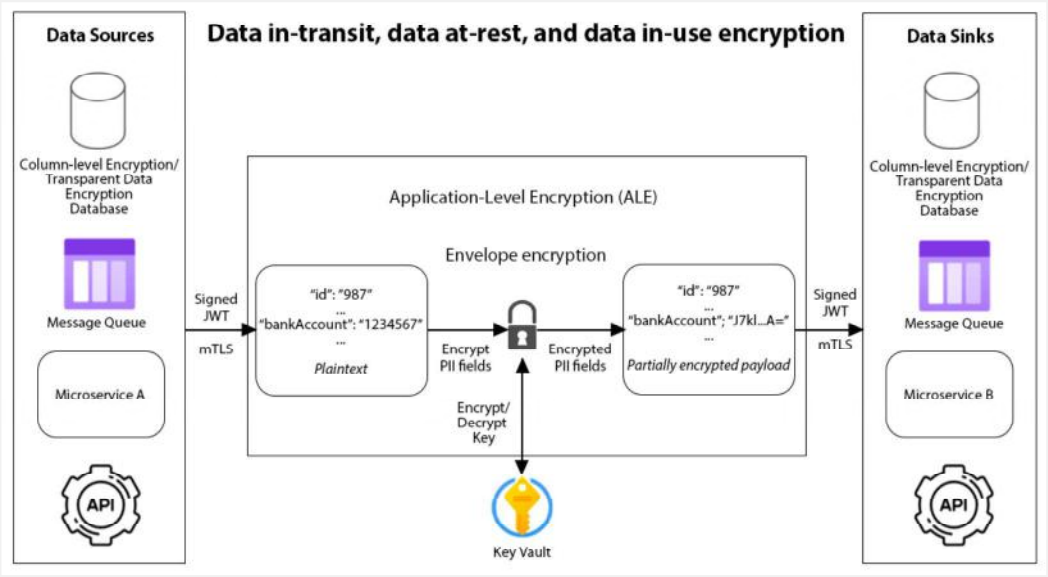
This reduces the blast radius of an attack and prevents attackers from pivoting from one compromised service into other parts of the infrastructure. In a container orchestration environment, such as Kubernetes, deifne network policies for egress and ingress isolation at a granular level. Enforce zero trust for all trafifc (east-west and north-south) by specifying network policies and service-to-service level RBAC policies that limit access per cluster and per source, following the need-to-know principle.

SECURE ACCESS TO RESOURCES

External entities must not access the microservices environment directly. Instead, use an API gateway as a single entry point to the microservices deployment. To pass the user context or the identity of the caller, implement a pattern, such as the phantom token pattern (*API Security in Action*, part 11.6.1) or the passport pattern. Validate the external access token and user context at the edge and generate a new short-lived token that represents the external entity identity and is cryptographically signed by the trusted issuer and propagated to back-end microservices. Ensure that the new token's scope of access is as limited as the scope of the identity of the external entity.

Most importantly, assume that access tokens can be stolen and create access tokens with a short lifespan on a resource-by-resource basis. Use a service mesh to verify the validity of the access tokens at the microservice edge. In all cases, access to resources should be granted using ifne-grained role-based access controls with the least privileges.

Figure 1: Data in-transit, data at-rest, and data in-use encryption



DATA SECURITY

It is essential to ensure that all data is classiifed according to their secrecy and conifdentiality. Create a data registry to know which microservice handles what data. Then, implement multiple layers of data encryption, depending on the data classiifcation. Do not trust only the encryption of external components (including databases and messaging systems like Kafka). Use application-level encryption (ALE) to transfer personally identiifable information (PII) and highly conifdential data between microservices. To mitigate the risk of unauthorized data modiifcation, perform data integrity checksums throughout the data lifecycle.

INFRASTRUCTURE SECURITY

Adopting an immutable infrastructure has become standard. Use Infrastructure as Code to provision components upfront and never change them after deployment. Do not trust the storage mediums (persistent or temporary) and do not store any sensitive data or secrets in an unencrypted form. All secrets, certiifcates, and API keys should be securely stored in access-controlled centralized key vaults.

Zero trust always assumes that the network is compromised. To contain a possible compromise and prevent lateral spreading through the rest of the network, implement network micro-segmentation, create software-deifned perimeters in each segment, and place microservices in each segment according to their functionality, business domain, and data classiifcation. Communication between segments should be well-deifned and controlled through API gateways. Consider adopting a cell-based architecture for inter-segment communication.

CONTAINER AND CLUSTER SECURITY

Zero trust requires the explicit veriifcation of container images, containers, and cluster nodes. Thus, use container images that are signed only from trusted issuers and registries. Allow images to be used only if they are scanned in the DevSecOps pipeline and have no vulnerabilities. To reduce the risk of privilege escalation, run the Docker daemon and all containers without root privileges. One standard way is to run Docker in rootless mode. Logically isolate high-risk applications and workloads in the same cluster for the least number of privileges.

RUNTIME SECURITY

Consider running security-sensitive microservices on conifdential virtual machines in hardware-based trusted execution environments with encrypted memory. To reduce the risk of rogue or compromised nodes in the cluster, verify the integrity of nodes, VMs, and containers by running them on instances enabled with Secure Boot and Virtual Trusted Platform Module.

Also, by running containers in read-only mode, iflesystem integrity is achieved and attackers are prevented from making modiifcations. Finally, we can reduce our trust for the runtime by adopting a RASP solution that inspects all code executed by the runtime and dynamically stops the execution of malicious code.

Figure 2: Zero-trust runtime via conifdential computing and RASP

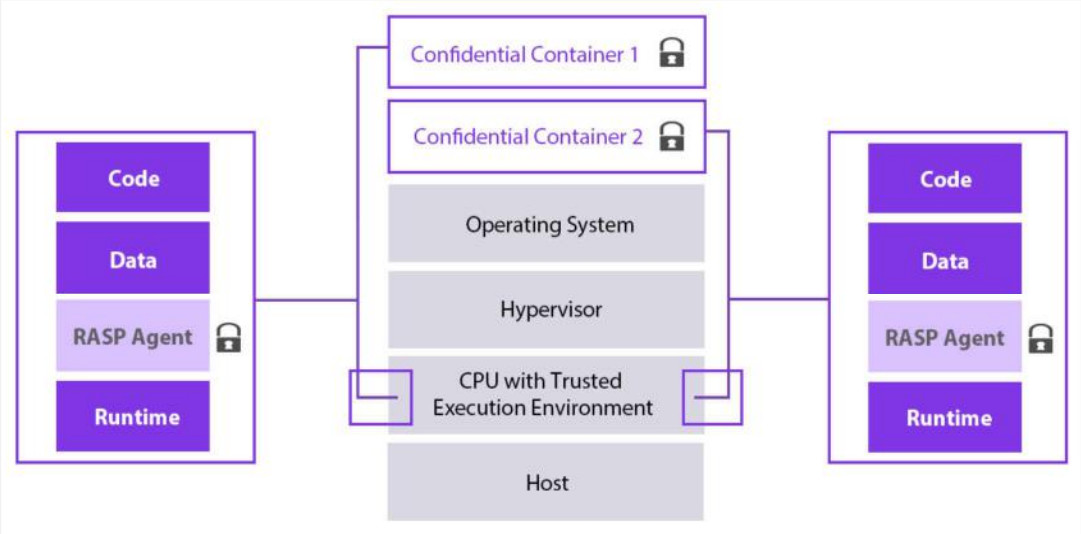


Image adapted from "Application enclave support with Intel SGX based conifdential computing nodes on AKS," Microsoft Azure Documentation

Conclusion

Implementing a zero-trust architecture is a critical defense-in-depth strategy and has become a mandatory security model in modern IT infrastructures. It is important to understand that implementing a zero-trust architecture does not mean zero security incidents. The goal is to continually layer security controls to increase the cost of attacks. As we introduce more friction into the cyber-attack kill chain, the attacker's value proposition will be reduced, and potential attacks will be disrupted.

The key to a successful implementation of a zero-trust architecture is to follow the guidance of whitepapers such as NIST's "Planning for a Zero Trust Architecture" and the U.S. Ofifce of Management and Budget's "Moving the U.S. Government Towards Zero Trust Cybersecurity Principles."

In this article, we provided an overview of how to apply the core principles of the zero-trust model in a microservices environment, and we examined the critical areas and the zero-trust security goals of microservices that need to be achieved. The highly distributed and heterogeneous nature of a microservice deployment and its complex communication patterns has increased the number of different components and the volume of data that is exposed on the network. This provides a broader attack surface compared to a traditional deployment of a monolithic application.

Because the security of a system is as good as its weakest link, applying the zero-trust core principles to proactively secure all layers and components of a microservices deployment is fundamental for a modern, reliable, and mature cybersecurity strategy. With a proper zero-trust strategy for microservices, the risk of compromised clusters, lateral movement, and data breaches in most cases can be eliminated.

Zero trust is a necessary evolution to security; however, its implementation should not be a destination. It is a continuous journey and an organization-wide commitment. Since its inception, zero trust has become a widely deployed security model and a business-critical cybersecurity priority. Microsoft's 2021Zero Trust Adoption Report conifrms that point on page 11, indicating that 76 percent of organizations have started adopting a zero-trust strategy. The industry is rapidly adopting zero trust across the whole infrastructure and not just on end-user access.

Apostolos Giannakidis, Principal Product Security Engineer at Microsoft

@agiannakidis on DZone | @giannakidisapostolos on LinkedIn | @cyberApostle on Twitter

Apostolos is passionate about all aspects of software and security. Before joining Microsoft, he worked as a Vice President of Application Security at JP Morgan Chase and drove Waratek's security strategy and research. Apostolos holds two MSc degrees in Computer Science and Cloud Computing. He has been acknowledged by Oracle and is featured on Google's Vulnerability Hall of Fame. He enjoys going to concerts, and his vinyl collection no longer ifts in his room.

CONTRIBUTOR INSIGHTS

Building a Secure Mobile App in the Cloud

A Complete Guide for iOS and Android

By Boris Zaikin, Software & Cloud Architect at Nordcloud GmbH

Building secure mobile applications is a dififcult process, especially in the cloud. We must consider that mobile platforms, like iOS and Android, have completely different architectures and quality guidelines. Also, we need to take care of our cloud architecture on the back end. In this article, we will have a look at the top six security vulnerabilities, OWASP's best practices for building/testing iOS and Android applications, and guidelines for iOS and Android. Last but not least, we will explore an example of DevSecOps for mobile applications.

Top Three Attack Examples

To understand the importance of security for mobile apps, let's ifrst look at three of the most prominent hacks of mobile apps that led to huge ifnancial and marketing issues for the affected companies.

PARKMOBILE BREACH

In the cyber attack on the ParkMobile app in 2021, hackers managed to steal 21 million user accounts. According to Security7, hackers managed to steal telephone numbers, license plate numbers, and email addresses. It seems like all the unencrypted data were stolen passwords. However, credit cards were encrypted, so hackers didn't manage to encrypt data as the keys weren't stolen.

JUSPAY DATA LEAK

Juspay, a payment operator that provides services for Uber, Amazon, Swiggy, and Flipkart, was hacked through their mobile app in August 2020. The hacker stole 35 million records, including credit card data, ifngerprints, and masked card data.

WALGREENS MOBILE APP LEAK

In 2020, Walgreens' mobile app had integrated malware that watched personal messages and info. It resulted in a lot of user data being compromised, including names, prescription numbers, and addresses.

Top Six OWASP Security Vulnerability Types in iOS and Android

Before we jump into iOS and Android guidelines and OWASP Testing Guides, let's look at the top six OWASP vulnerability types:

Table 1

Vulnerability Type	Description
Authentication issues, insecure communication	A mobile application has unencrypted UI forms, algorithms, and protocols to authenticate. An attacker uses the fake app/malware to scan and observe the application transport layer. Also, weak passwords, using geolocation to authenticate users, or using persistent authentication may lead to sensitive data leaks.
Reverse engineering	This vulnerability allows an attacker to analyze and obfuscate the targeted application. This may lead to sensitive data leakage that is hard coded in application conifguration variables or constants. In addition, attackers may ifnd URLs and conifgs to the back-end servers.
Data storage security vulnerability	This vulnerability allows attackers to steal data from data storage. We partially link it with "improper platform usage." To prevent data leakage, we should use only encrypted data storage, avoid storing sensitive data (passwords, card numbers) in the device, encrypt data transfer, and use only encrypted storage OS features (e.g., iOS Keychain). We can reference CWE-922 of the mobile vulnerability registry.

TABLE 1 CONTINUES ON NEXT PAGE

Table 1 (continued)

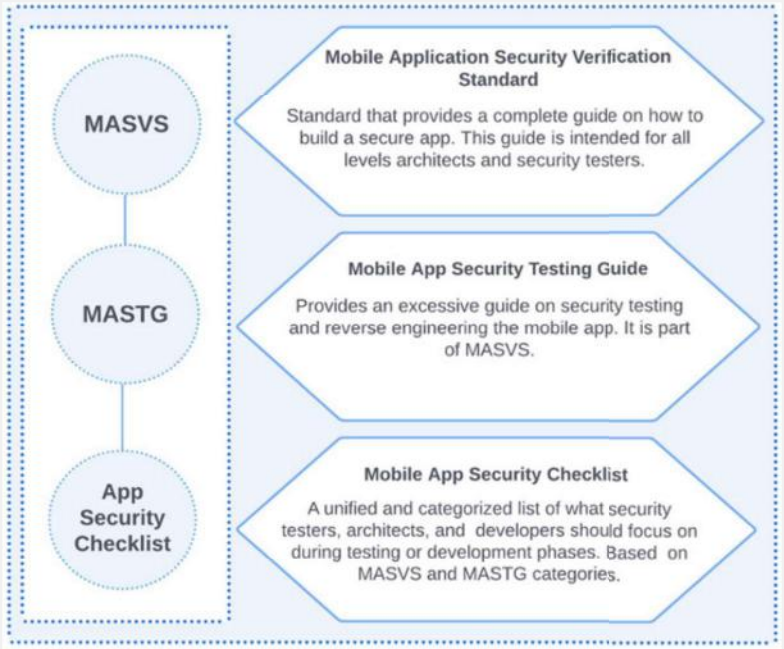
Vulnerability Type	Description
Improper platform usage	This type of attack relies on the issue of developers not using (or improperly using) security features that are included in the operation system. Security features include Face ID, iOS Keychain, and Touch ID. For example, developers may use insecure local storage instead of iOS Keychain to store sensitive data.
Code tampering	Code tampering is when an attacker downloads an app and makes code changes. For example, they create fake registrations or payment forms and then upload apps back to the market or create cloned ones. It can also be a fake app (such as free mobile cleaning tools or free games in app stores) that can modify the code of another app. Usually, banking apps are one of the scenarios to target, and Mobile ZeuS or Trojan-Spy can be used to steal mobile TAN code.

In my opinion, this is a list of the most important vulnerability types. However, OWASP provides a list of 10, and it also provides standards and testing guides. We will cover these in the next section.

OWASP Mobile Application Security Fundamentals

OWASP mobile application security fundamentals consist of several sources and contain OWASP Mobile AppSecurity Veriifcation Standard (MASVS), OWASP Mobile Application Security Testing Guide (MASTG), and the Mobile Security Checklist. Below in Figure 1, you will see the fundamentals of mobile application security in detail:

Figure 1: OWASP mobile app security fundamentals



Let's have a more detailed look at the mobile app checklist.

MOBILE APPLICATION SECURITY CHECKLIST

The Mobile Application Security Checklist is a part of the MASTG. It is a set of rules/checks that a dev team should include when securing a mobile app. It contains more than 100 rows and is organized by the following categories:

- Architecture, Design, and Threat Modeling Requirements
- Data Storage and Privacy Requirements
- Cryptography Requirements
- Authentication and Session Management Requirements
- Network Communication Requirements
- Platform Interaction Requirements
- Code Quality and Build Setting Requirements
- Resilience Requirements

Each rule (or check) has an identiifcation code and description. All rules have priority marks. "L1" or "L2" means that the application should have the rule/check implemented. "R" means that it is required, so the team must implement everything marked "R." Download the full example on OWASP's website.

Next, let's focus on guidelines for speciifc platforms, with attention to the most popular ones: iOS and Android.

Secure Mobile Apps in iOS and Android: Guidelines

As we have already partially touched some iOS security APIs, we will continue discussing it with the addition of Android. In the ifrst section below, I've gathered guidelines and best practices about iOS API security features.

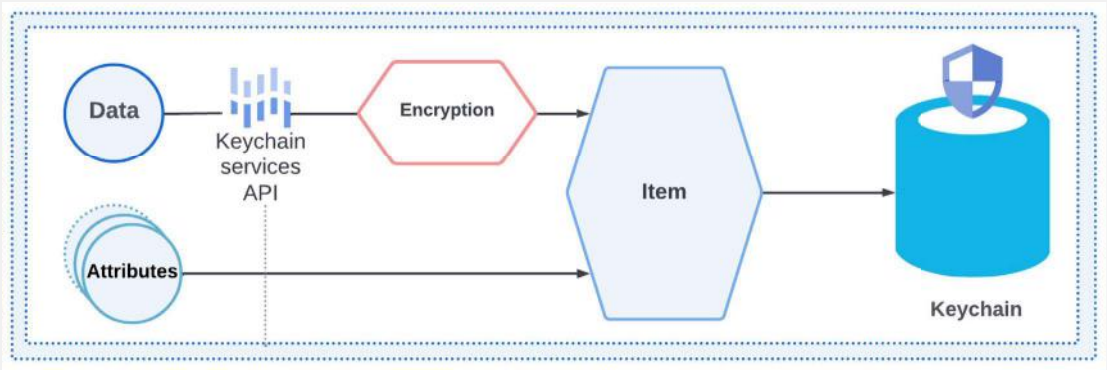
APPLE APP SANDBOX, DATA PROTECTION API, AND KEYCHAIN

The Apple App Sandbox provides an API to isolate an app and prevent access to the main system or other apps. It's based on UNIX's user permission and ensures that apps get executed with a less privileged "mobile" user. Also, it includes address space layout randomization (ASLR) and ARMs Never eXecute, which prevent memory-related security bugs and stops malicious code from being executed.

The Data Protection API allows an app to encrypt and decrypt its ifles, and it may solve several security issues like authentication and reverse engineering. Each ifle has four available protection levels, and by default, it's encrypted with the ifrst user authentication. However, we should increase the level to provide the highest protection.

Last but not least, the keychain. It provides secured hardware-accelerated data storage. iOS provides this API to store certiifcates and passwords with the highest level of security. For each item in the keychain, we can deifne speciifc access policies. Especially when the user needs to request Face ID or Touch ID, the biometric enrollments won't change since the item was added to the keychain.

Figure 2: Keychain API



ANDROID-ENCRYPTED KEY-VALUE STORAGE, FILE ENCRYPTION, AND CRYPTOGRAPHIC APIS

Same as iOS, Android also has many similar features to store data securely. The ifrst one is key-value storage. It allows storing data using SharedPreferences to set a scope of visibility for items in the storage. We need to keep in mind that stored values are not encrypted by default. Therefore, malware may have access to the data.

If we need to encrypt data manually, we beneift from using Cryptographic API. We can generate a secure key with KeyGenerator, then save and extract the encrypted value to Android Keystore. To work securely with ifles and external storage, Android has the Cryptography Support Library. It supports a lot of cryptography algorithms to encrypt/decrypt ifles.

HTTPS, SSL PINNING, AND PUSH NOTIFICATIONS

A secured communication layer is the next big milestone to a secured app. First, we need to ensure that we are using HTTPS. iOS has a feature called App Transport Security (ATS) that blocks insecure connections by default, so all connections must use HTTPS/TLS. In addition, the SSL pinning feature helps to prevent man-in-the-middle attacks. It will validate the system certiifcate if it were signed by a root certiifcate authority.

To use this feature, the app should run additional trust validation of server certiifcates. Push notiifcations are another part that should be secured. We should use Apple's Push Notiifcation service (APNs) and the UNNotificationServiceExtension extension. This will allow us to use placeholders for sensitive mobile app data and send encrypted messages.

Also, consider using Apple's CryptoKit. It is a new API introduced in iOS 13 that provides the following features:

- Hashing data
- Authenticating data using message authentication codes
- Performing key agreement
- Creating and verifying signatures

Android has similar options. It allows only HTTPS to transport encrypted data with TLS. And it is the same story for SSL pinning. To prevent man-in-the-middle attacks, we can perform additional trust validations of the server certiifcates.

Secure Mobile Apps in Azure and AWS

To build secure applications, Azure has services such as the Azure App Center. It allows for the building and distribution of mobile apps and provides a lot of security options:

- Data transit encryption – support HTTPS using TLS 1.2 by default; also encrypted at rest
- Code security – provides multiple tools to analyze code dependency to detect security vulnerabilities
- Authentication – contains features like Microsoft Authentication Library (MSAL), which supports multiple authorization grants and associated token lfows

Alongside Azure, AWS has some powerful services to consider when building a secure mobile app. Take AWS Cognito as an example. It is a user-state service with options to develop unique identities for users. It supports:

- Secure app authentication
- Enabling developers to include user sign-up
- Easy sign-in and access control focused on web and mobile apps

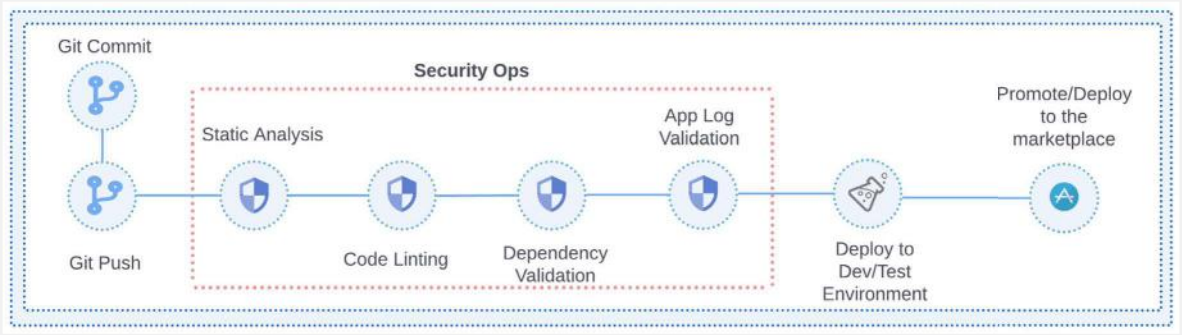
AWS has one unique service with the name AWS Device Farm. It provides not only automated testing and simulation environments, but also contains features to validate app dependencies and run security checks. Now let's move on to an example of building a DevOps process with security features.

An Example of DevSecOps for Mobile Applications

In this section, I've created an example of a DevSecOps scenario to deliver secure mobile applications (see Figure 3). This process can be reused within the most popular CI/CD platforms and cloud providers:

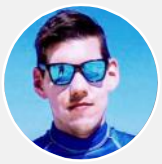
1. **Git operation steps** – Contains standard commit/push operations when the source control triggers a build.
2. **Run static analyses and code linting steps** – Validates the code styles, usability, data lfow issues, and security issues (e.g., Xcode Static Analyzer).
3. **Dependency validation step** – Provides excessive validation checks through the library tree used in the app. This validation step may reveal a fake, malicious library that can manipulate code or even steal personal user data.
4. **Application log validation step** – Checks if logs contain sensitive data like environment passwords, test tokens, or authorization data. After the dev/test process, the application package may contain some sensitive data as developers may not notice it after debugging the app. (This step can be run after deployment to the dev/test environment as well).
5. **QA steps:**
   - Deploy the app to the dev/test environment for the QA team to test.
   - Promote and deploy the app to the marketplace validation.

Figure 3: Common DevSecOps process of a secure mobile app



Conclusion

In this article, I've provided a short guide on secure mobile applications. We discovered that the OWASP community has major security fundamentals, and OWASP can be used as a strong base for building a new app or refactoring an existing one. Knowing cloud services and examples of DevSecOps allows us to start building secure mobile apps with minimum effort and makes it harder for an attacker to compromise our app. Also, we went through iOS and Android security features, security APIs, and discovered how to use them properly.



Boris Zaikin, Software & Cloud Architect at Nordcloud GmbH

@borisza on DZone | @boris-zaikin on LinkedIn | boriszaikin.com

I'm a certiifed senior software and cloud architect who has solid experience designing and developing complex solutions based on the Azure, Google, and AWS clouds. I have expertise in building distributed systems and frameworks based on Kubernetes and Azure Service Fabric. My areas of interest include enterprise cloud solutions, edge computing, high-load applications, multitenant distributed systems, and IoT solutions.

CONTRIBUTOR INSIGHTS

Surviving the Incident

Building a Tailored Guide for Handling Security Incidents

By Roderick Chambers, Senior Information Security Specialist at Recorded Future

A wave of cyber incidents in recent years, such as the SolarWinds supply chain attack, Accellion data breach, Exchange Server, and Log4j vulnerabilities, have exposed the "fragility" of modern businesses and the challenges in information security. The outcome of a cyber event can range from a minor business operations disruption to "crippling ifnancial and legal costs" (Alan P., GCST). Despite how increasingly sophisticated threat actors have become, the typical attack scenarios remain the same, and thus analysts and responders can adequately address these events using previous tactics.

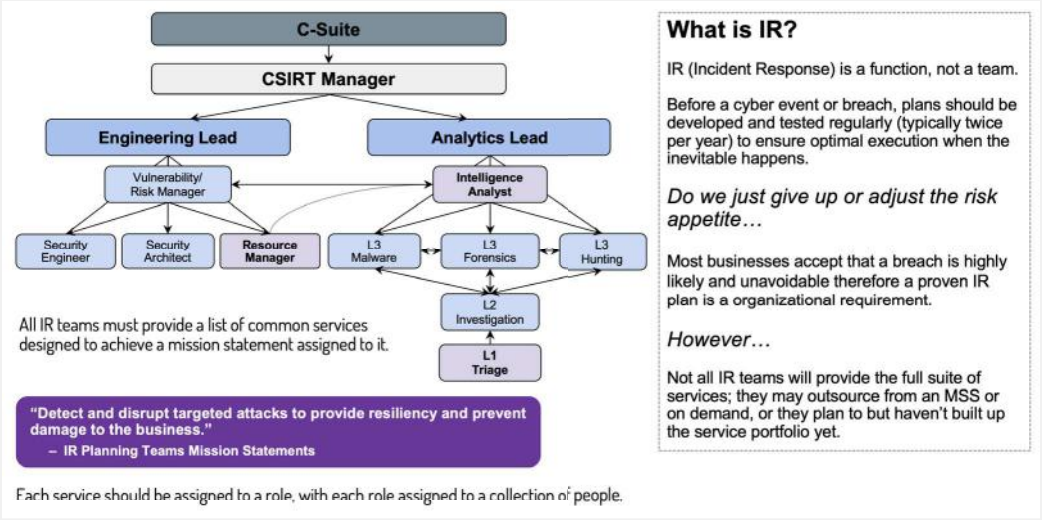
Effective incident response (IR) playbooks are the best antidote to unpredictable attacks. A practical IR playbook is key to stay organized and lead to minimal risk. According to Jyotsana Gupta, an "incident response is a process that allows you to respond quickly and effectively to a cybersecurity breach" (Wire19). IR playbooks enable analysts to respond to an incident consistently, ensure correct procedures are followed, and provide organizations with a roadmap to determine where processes can be automated and enhanced to improve critical response time.

What goes into creating an IR playbook? We will discuss how to design an IR playbook for an organization, covering topics such as assembling your IR team, identifying critical systems, creating notiifcation procedures, and conducting post-incident review. For those eager to design a large-scale IR playbook, NIST and SANS are the two most popular incident response frameworks for granular IR planning approaches. While they differ in categorizing the incident response phases, both follow the same basic process.

How to Build an Incident Response Playbook

What's your plan? While Indiana Jones might say, "I don't know, I'm making this up as I go," good luck trying that line on customers and regulators. To begin drafting the IR playbook, some companies require engagement from critical business units to form an incident response (IR) team, while other large organizations may already have an established, dedicated computer security incident response team (CSIRT). The core of the IR or CSIRT team will usually be IT or cybersecurity staff, and the ideal sponsor is from the C-Suite, such as the CISO, who will help empower the team with resources to act swiftly and drive accountability across the organization. Other members may be drawn from the IT operations staff, the vulnerability and risk management team, security engineers and architects, and intelligence analysts. Extended partners may include other capabilities, such as PR, HR, and legal.

Figure 1: Hierarchy of an CSIRT/IR team



IDENTIFY THE CROWN JEWELS

The next step to the IR playbook is to identify the "crown jewels" of the organization — the critical systems, services, and operations that, if impacted by a cyber event, would disrupt business operations and cause a loss of revenue. Similarly, understanding the collected data type, how it is transmitted and stored, and who should access it must be mapped to ensure data security.

Identifying and mapping critical systems can be accomplished through penetration tests, risk assessments, and threat modeling. A risk assessment is often the ifrst tool to identify potential attack vectors and prioritize security events. However, to achieve a proactive stance, organizations are increasingly leveraging threat intelligence and modeling to identify and address vulnerabilities and security gaps early on before a known attack occurs. The primary goal is to identify weaknesses or vulnerabilities with assets to reduce the attack surface and close all the security gaps.

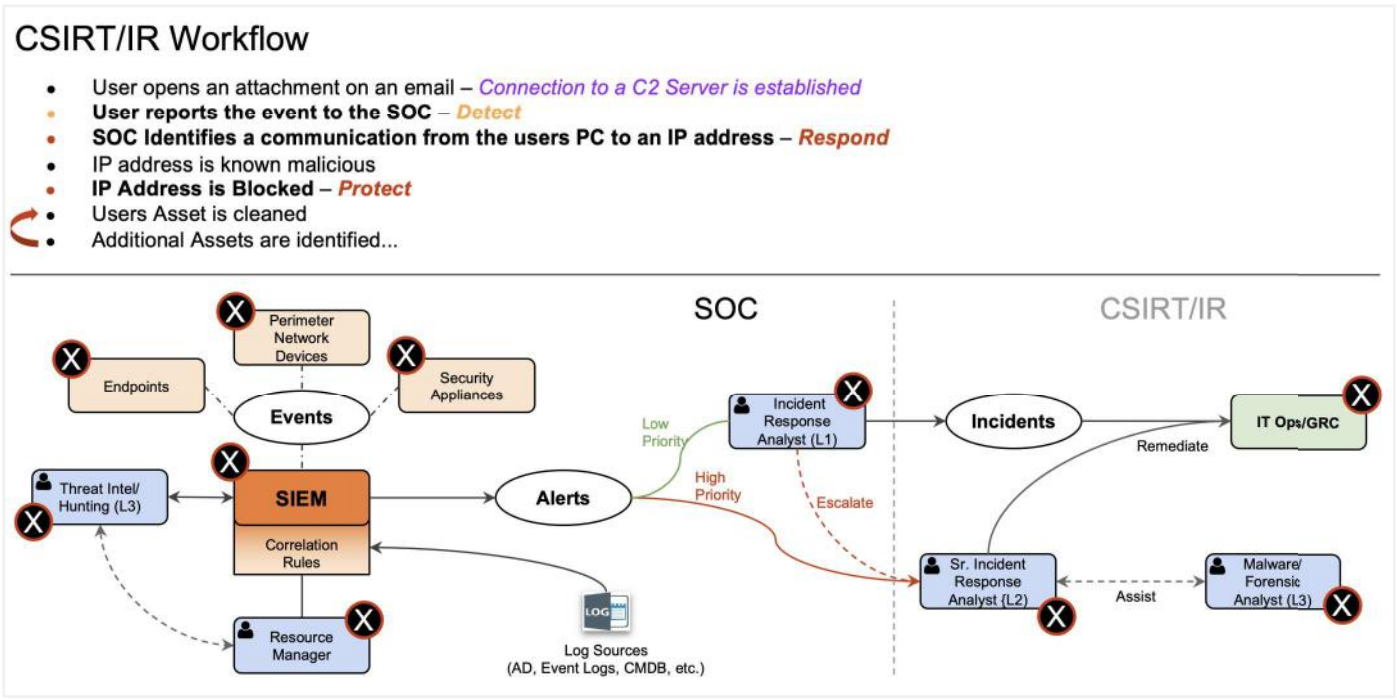
This guide will focus on web application security as our attack scenario. Why web application security? Applications have become the backbone of most organizations, making web applications, websites, and web-based services a prime target for malicious threat actors attempting to exploit vulnerable application code. As is stated by Gupta:

> *Web applications are attractive targets for the following reasons:*
>
> - ***Complexity –*** *web applications have inherent complex source code, making it likely that an app contains unpatched vulnerabilities or is open to code manipulation.*
> - ***High-value rewards –*** *attackers can manipulate source code to access valuable data, including personal and sensitive business information.*
> - ***Easy execution –*** *attacking web applications is usually straightforward with automation and large-scale attacks targeting multiple sites simultaneously (Wire19).*

Understand the Threats

The next step is creating notiifcation procedures led by the IR or CSIRT team. In the attack scenario (Figure 2), the IR team or CSIRT will often begin with basic security procedures of securing web applications, such as deploying and conifguring a web application ifrewall (WAF) or conifguring access control policies. The security team might block cyber attacks using WAF rules to block active exploits and prevent further damage.

Figure 2: Sample attack scenario response worklfow



Web application conifgurations allow and deny access by creating policy rules in the web access layer. A baseline example is focusing on URL category ifltering by creating policies:

- Allow users to access all organizations' approved social networking sites such as LinkedIn and block other sites such as TikTok.
- Allow users to access personal email accounts but prevent sending email attachments.

When WAFs and conifgurations are in place, the web application is monitored for suspicious activity. The communication aspect of the IR playbook comes into play when careful logging and monitoring at the application level detect suspicious activity, such as repeated access attempts or unexpected user accounts being created. Suspicious activity is often detected using a security information and event management (SIEM) solution or through regular security testing using a web vulnerability scanner (Alan P., GCST).

After detecting a security incident, the IR team should "triage it," meaning they should determine the appropriate action to limit short-term consequences and stop minor incidents from growing into large-scale attacks (Alan P., GCST). But what happens when the cyber event cannot be contained, and the evidence of a data breach is quickly mounting?

According to Alan P., "global cyberattacks have served as a reminder that plugging security holes is often the easiest part." Threat actors such as advanced persistent threats (APT) don't conduct hit-and-run attacks. These attacks "inifltrate target systems to maintain a stealthy and persistent presence. Eliminating the entry point is the start of a long and arduous process" with the IR or CSIRT team (Alan P., GCST).

DETERMINE COMMUNICATION CHANNELS

When outages in programs or degradation of system performance occur, communication must be clear and effective. Communication will directly impact your team's responsiveness to threats. While the CSIRT team is focused on analysis, containment, and remediation, proper incident response communications to leadership and read-in members are critical to success.

Managers should be informed of the situation and understand the implications to give their team the next steps to respond to an incident. Many users have questions such as whether the users should keep working or turn off their machines. What is worse are users unplugging machines with the belief it will stop the issue from spreading (David Landsberger, CompTIA).

The next phase is to manage staff and customers. Gupta notes:

> *After a data breach, there might be a legal obligation to notify data owners (i.e., per the GDPR). The jurisdiction and data class will determine whether you must report the incident to the relevant authorities and provide updates when new information is available (Wire19).*

A few examples of how to communicate an incident to staff and customers include:

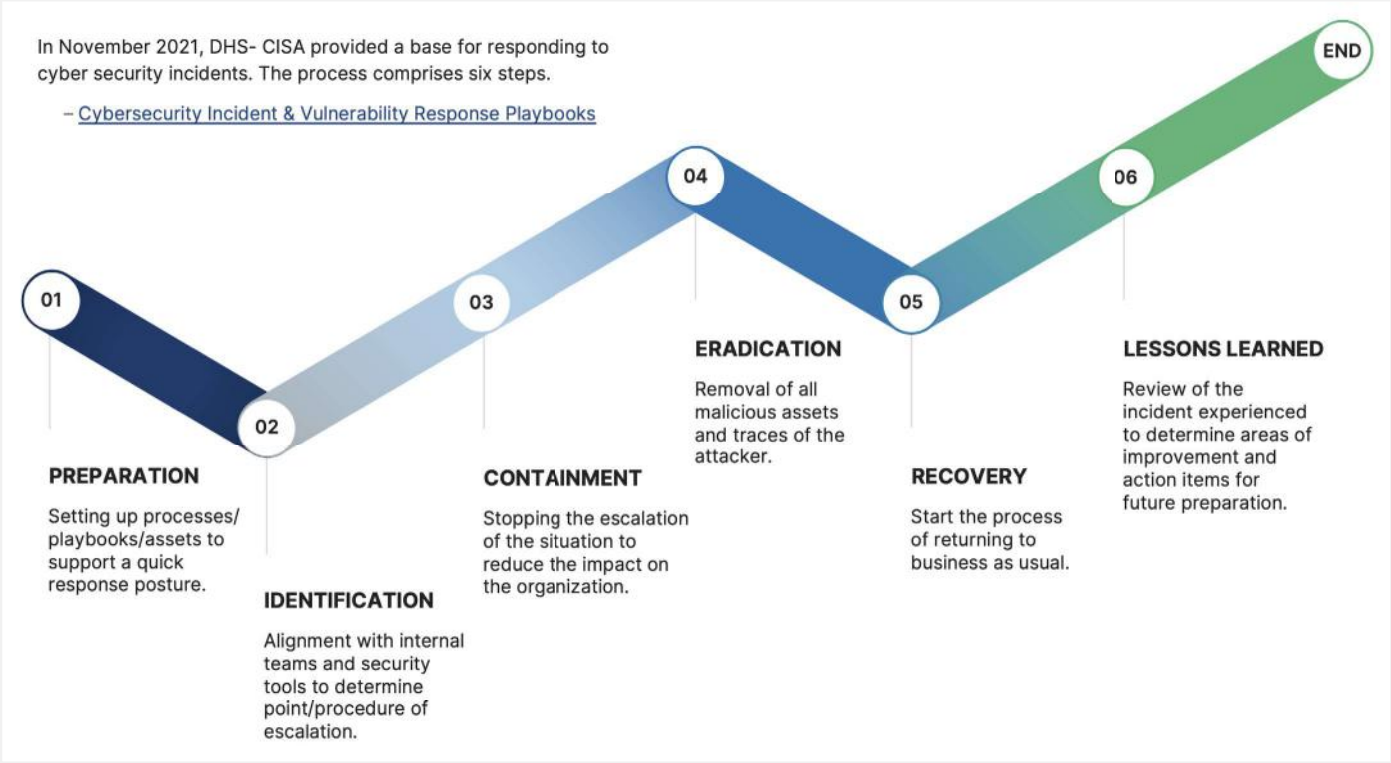
- An email for internal incident response
- A status page
- A message via chat channels within your team (e.g., Slack)
- Outward-facing communication with clients and customers through public relations teams

ENABLE POSTMORTEMS

Molly Star (Lightstep Blog) states: "Incident response management doesn't end when the incident is resolved. Your playbook should detail your postmortem process, from documentation and discussion to action items." Postmortem follow-up is essential because the organization needs to review how they can improve the security of their systems and, in other words, "harden" their perimeter.

The ifrst step is to identify the root cause of the cyber event via the logs on your ifrewall/WAFs or through the SIEM. By identifying the root cause, remediation can begin to prevent the same or similar cyber events. No solution is perfect, and even if an organization is doing everything right, it is best to adopt an assume-breach mentality.

Figure 3: Summary of incident response process



Conclusion

Data breaches are inarguable and demonstrate a failure to secure systems, and it's on the organization to respond quickly and effectively. Having a tested IR playbook in hand, continually practiced through tabletop exercises, can help your team cross the ifnish line while minimizing material impact to the company. Incidents can have a lasting impact, even if adequately handled. A developer-driven, security-ifrst mindset is worth considering. It has a positive ripple effect on the business if developers and security teams can work together and learn lessons from previous incidents.

This guide is a wake-up call to implement holistic information security practices that have both developers and security teams moving toward the same goal as a collective. Dedicating time to threat model with developers to build secure code upfront is a small resource investment that has the potential to yield signiifcant gains for the business in the long term. After all, it can shrink the attack surface, reducing impact should an incident occur.

References:

- "An Incident Response Plan for Your Website" by Jyotsana Gupta (Wire19)
- "What Is an Incident Response Plan and How to Create One" by David Landsberger (CompTIA)
- "Incident Response Steps in Web Application Security" by Alan P. (GCST)
- "Building an Incident Response Playbook" by Molly Star (Lightstep Blog)

Roderick Chambers, CISM, CISSP, Senior Information Security Specialist at Recorded Future
@roderi on DZone | @roderickchambers1on LinkedIn

Chambers serves as public and private sector entities' information security and intelligence advisor. He began his career in the US federal government, speciifcally in the intelligence community, where he served as intelligence operations professional and technical collections lead. Chambers served as the former deputy superintendent and director of the cyber intelligence unit for the US State of New York State Department of Financial Services.

ADDITIONAL RESOURCES

Diving Deeper Into Application Security

MULTIMEDIA

Cloud Security Podcast

This vendor-neutral and easily accessible weekly podcast is hosted by Ashish Rajan and Shilpi Bhattacharjee. You can catch them discussing all aspects of cloud security on your favorite podcast platform, but be sure to catch their podcasts live streaming on YouTube and Twitter every weekend.

Application Security Weekly

Hosts Mike Shema, John Kinsella, and Akira Brand interview industry professionals from the AppSec, DevOps, DevSecOps, and other related ifelds to give you all the news and information you need to know in these realms. Their podcast will help you make sure your modern security practices are strong enough to ifght off any attack.

Open Source Security Podcast

This security podcast is like no other: It talks about open source in every episode. Join hosts Kurt Seifried and Josh Bressers to learn more about popular security topics that span application security, IoT, cloud, DevOps, and more. With more than 300 episodes, you're bound to ifndthe information you want and the information you didn't know you needed.

Security Now

Steve Gibson and Leo Laporte will keep you abreast on all the news and hot topics you need to know in order to stay up to date in the security realm. You can catch them streaming live every Tuesday or browse their well-stocked library of podcasts. And a little fun fact for you: Gibson coined the term "spyware" and created the ifrst anti-spyware program!

@TCMSecurityAcademy

From easy-to-digest 10-minute videos to two-hour courses, The Cyber Mentor YouTube Channel covers topics such as web application penetration testing, Linux, bug hunting reviews, tools, and more. TCM tackles everything practically but with a bit of fun, keeping you constantly entertained.

REFCARDS

Cloud-Native Application Security: Patterns and Anti-Patterns

Cloud-native applications leverage modern practices like microservices architectures, containerization, DevOps, IaC, and automated CI/CD processes. This Refcard will walk through the critical challenges of cloud-native security, show how to build security into the CI/CD pipeline, and introduce the core patterns and anti-patterns of cloud-native application security.

IaC Security

The responsibility and accountability for security is rapidly shifting toward DevOps engineers, as they have greater visibility into the broader architecture of processes and systems used to deploy applications. Effective DevSecOps makes app deployments, operations, and service monitoring easier and more secure. In particular, DevOps engineers will be responsible for securing the Infrastructure as Code in which they build. In this Refcard, we explore IaC security, how it works, why it's important, and core practices for success.

Introduction to DevSecOps

With DevSecOps, you can reach higher security standards while following DevOps principles. This Refcard will show you how to get started with DevSecOps with key themes, crucial steps to begin your journey, and a guide to choosing security tools and technologies to build your DevSecOps pipeline.

TREND REPORTS

Application Security

In DZone's 2021 Application Security Trend Report, readers will discover how the shift in security focus across the SDLC is impacting development teams — from addressing the most common threat agents and attack vectors to exploring the best practices and tools being employed to develop secure applications.

DevSecOps

Security has long been an afterthought, but organizations have started incorporating security into their DevOps pipelines. With this in mind, we consulted industry experts and leaders about the state of DevSecOps adoption and implementation to help readers understand more effective ways to manage security throughout every step of the SDLC.

ADDITIONAL RESOURCES

Solutions Directory

This directory contains authentication, vulnerability detection, application security testing, IAM, WAF, cloud security, DDoS protection, penetration testing tools, as well as many other tools to assist you with your application security. It provides pricing 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.

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

2022 PARTNERS

Company	Product	Purpose	Availability	Website
Azul	Azul Vulnerability Detection	Agentless cloud solution to identify and track Java vulnerabilities	Free tier	azul.com/products/vulnerability-detection
Check Point Software Technologies Ltd.	CloudGuard CNAPP	Cloud native application protection	By request	checkpoint.com/cloudguard/cnapp
Zimperium	zDefend	Runtime Application Self-Protection (RASP)	By request	zimperium.com/zdefend
	zKeyBox	Cryptographic key protection		zimperium.com/zkeybox
	zScan	App security testing		zimperium.com/zscan
	zShield	App shielding/obfuscation		zimperium.com/zshield
Company	**Product**	**Purpose**	**Availability**	**Website**
A10 Networks	Thunder CFW	Convergent ifrewall and DDoS protection	By request	a10networks.com/products/thunder-cfw
	Thunder SSLi	SSL visibility and decryption		a10networks.com/products/thunder-ssli
	Thunder TPS	DDoS detection and mitigation		a10networks.com/products/thunder-tps
	Thunder ADC	Application delivery and load balancing		a10networks.com/products/thunder-adc
	Thunder Harmony Controller	Service analytics and management		a10networks.com/products/harmony-controller
Acunetix by Invicti Security	Acunetix	Web application security testing	By request	acunetix.com
Airlock	Airlock	Secure access management	By request	airlock.com/en
Akamai	App & API Protector	Security for websites, apps, and APIs	Trial period	akamai.com/products/app-and-api-protector
	Enterprise Application Access	Zero-trust network access		akamai.com/products/enterprise-application-access
	Secure Internet Access Enterprise	Secure web gateway		akamai.com/products/secure-internet-access-enterprise
Amazon Web Services	AWS WAF	Web application protection	Free tier	aws.amazon.com/waf
	Amazon CloudFront	Low-latency content delivery network		aws.amazon.com/cloudfront
	Amazon Cognito	Customer identity and access management		aws.amazon.com/cognito
	AWS Shield	Managed DDoS protection		aws.amazon.com/shield
	AWS Secrets Manager	Secrets lifecycle management		aws.amazon.com/secrets-manager

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
Anchore	Anchore Enterprise	SBOM-powered, end-to-end software supply chain security	By request	anchore.com/platform
Anomali	Anomali Platform	Extended detection and response	By request	anomali.com/products
Aqua	Aqua Trivy	Vulnerability and misconifguration scanning	Open source	aquasec.com/products/trivy
	Aqua kube-bench	Kubernetes deployment security		github.com/aquasecurity/kube-bench
	Aqua CloudSploit	Cloud security posture management		github.com/aquasecurity/cloudsploit
	Aqua Platform	Cloud-native security	By request	aquasec.com/aqua-cloud-native-security-platform
Armor	Armor	Cloud-native cybersecurity	By request	armor.com
Arnica	Arnica	Software supply chain security automation	By request	arnica.io
AT&T Cybersecurity	AT&T Secure Web Gateway	Location, user, and device uniifed protection	By request	cybersecurity.att.com/products/secure-web-gateway
	AT&T DDoS Defense	Cloud-based monitoring of volumetric DDoS attacks		cybersecurity.att.com/products/reactive-ddos-services
	AT&T Token Authentication Service	User authentication service		cybersecurity.att.com/products/token-authentication
	AT&T Managed Endpoint Security with SentinelOne	Endpoint protection, detection, response, and control		cybersecurity.att.com/products/sentinel-one
Barracuda	Cloud Application Protection	Web application and API protection	Trial period	barracuda.com/products/application-cloud-security
	CloudGen Firewall	Distributed network security and optimization		barracuda.com/products/cloudgenifrewall
BeyondTrust	Endpoint Privilege Management	Least privilege enforcement	By request	beyondtrust.com/privilege-management
	Secure Remote Access	Remote access management		beyondtrust.com/secure-remote-access
Bitdefender	GravityZone Business Security Enterprise	Endpoint protection, detection, and response	Trial period	bitdefender.com/business/products/gravityzone-enterprise-security.html
	GravityZone Security for Mobile Devices	Mobile device protection for organizations		bitdefender.com/business/enterprise-products/mobile-security.html
BlackBerry Cylance Endpoint Security	Cylance Endpoint Security	AI-driven endpoint security	By request	blackberry.com/us/en/products/cylance-endpoint-security
BMC	BMC AMI Security	Automatic mainframe threat detection and response	Trial period	bmc.com/it-solutions/bmc-ami-mainframe-security.html
Broadcom	ACF2™	Scalable modern mainframe security	By request	broadcom.com/products/mainframe/identity-access/acf2
	Symantec Enterprise Cloud	Data-centric hybrid security		broadcom.com/products/cybersecurity
Browser Exploitation Framework Project	BeEf	Penetration testing	Open source	beefproject.com
Cavirin	Hybrid Cloud Security & Compliance Platform	Real-time monitoring, threat detection, and auto-remediation	By request	cavirin.com/products.html

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
CDNetworks	Application Shield	Cloud-based web service protection	Trial period	cdnetworks.com/cloud-security/application-shield
	Flood Shield	Cloud-based DDoS protection		cdnetworks.com/cloud-security/lfood-shield
Chainguard	Chainguard Enforce	Software supply chain risk management	By request	chainguard.dev/chainguard-enforce
Checkmarx	Checkmarx One	Comprehensive application security	Trial period	checkmarx.com/product/application-security-platform
	Checkmarx SAST	Static application security testing	By request	checkmarx.com/cxsast-source-code-scanning
	KICS	Static code analysis of IaC	Open source	checkmarx.com/product/opensource/kics-open-source-infrastructure-as-code-project
	Checkmarx SCS	Supply chain security	By request	checkmarx.com/cxscs-supply-chain-security
	Checkmarx SCA	Open-source risk scanning		checkmarx.com/cxsca-open-source-scanning
CIRT.net	Nikto2	Web server scanning and testing	Open source	cirt.net/Nikto2
Cisco	Secure Endpoint	Endpoint protection across control points	Trial period	cisco.com/site/us/en/products/security/endpoint-security/secure-endpoint/index.html
	Umbrella	Cloud-delivered security		umbrella.cisco.com
	Secure Cloud Analytics	Uniifed threat detection for on-prem and cloud environments		cisco.com/c/en/us/products/security/stealthwatch-cloud/index.html
Citrix	App Delivery and Security Service	Application performance and security	By request	citrix.com/products/citrix-app-delivery-and-security
Cloudentity	Cloudentity	Authorization, API access, and data security	Free tier	cloudentity.com
Cloudflare	Cloudflare Application Security Portfolio	Web application and API protection	Free tier	cloudflare.com/application-security
Code42	Incydr	Data leak and IP theft protection	By request	code42.com/incydr
Codenotary	Trustcenter	SBOM management	Trial period	codenotary.com/products/trustcenter
Cofense	Cofense Protect	Phishing attack detection	By request	cofense.com/product-services/cofense-protect
	Cofense Triage	Phishing email analysis, identiifcation, and mitigation		cofense.com/product-services/cofense-triage
Conviso	Conviso Platform	DevSecOps	By request	convisoappsec.com
Corelight	Investigator	Network detection and response	By request	corelight.com/products/investigator
CrowdStrike	Falcon Platform	Endpoint, cloud workload, identity, and data security	Trial period	crowdstrike.com/falcon-platform
Cybeats	SBOM Studio	SBOM lifecycle management with cybersecurity insights	By request	cybeats.com/sbom-studio
	RDSP IoT Security	IoT and connected device security		cybeats.com/rdsp-iot-security
CyberArk	Conjur	Secrets management	Open source	conjur.org

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
Cybereason	Cybereason Defense Platform	AI-driven security to detect, expose, and respond to threats	By request	cybereason.com/platform
Cycode	Cycode	Supply chain security	By request	cycode.com
Darktrace	PREVENT	Attack surface management	Trial period	darktrace.com/products/prevent
	DETECT	Threat detection		darktrace.com/products/detect
	RESPOND	Preemptive cyber attack response		darktrace.com/products/respond
Deepfactor	Deepfactor Developer Security	Vulnerability, supply chain risk, and compliance violation discovery and resolution	By request	deepfactor.io
Digital.ai	Application Security	Application monitoring and protection	By request	digital.ai/products/application-security
DigitalStakeout	DigitalStakeout	OSINT data analysis and alerting platform	By request	digitalstakeout.com/platform
DXC Technology	Cyber Defense	Attack detection, response, and remediation	By request	dxc.com/us/en/services/security/cyber-defense
Edgescan	Smart Vulnerability Management Platform	Full-stack coverage and attack surface management	By request	edgescan.com/platform
Elastic	Elastic Cloud	Security, observability, and enterprise search	Trial period	elastic.co/cloud
ESET	PROTECT Platform	Cybersecurity ecosystem	Trial period	eset.com/us/business/protect-platform
	ESET Cyber Security	Cybersecurity protection for macOS		eset.com/us/home/cyber-security
ExtraHop	Reveal (X) 360	SaaS-based network detection and response	Trial period	extrahop.com/products/cloud/how-it-works
F5	DDoS Hybrid Defender	Network and application DDoS protection	By request	f5.com/products/security/ddos-hybrid-defender
	Distributed Cloud DDoS Mitigation	Network, SSL, and application-targeted attack detection and mitigation	Free tier	f5.com/cloud/products/l3-and-l7-ddos-attack-mitigation
	Distributed Cloud WAF	Distributed web application protection		f5.com/cloud/products/distributed-cloud-waf
	Distributed Cloud API Security	Data leak and API protection	By request	f5.com/cloud/products/api-security
Fastly	Next-Gen WAF	Application and API protection and security	By request	fastly.com/products/web-application-api-protection
Fidelis Cybersecurity	Fidelis Elevate Platform	Active extended detection and response	By request	ifdelissecurity.com/platforms/elevate
	Fidelis Halo	Cloud security and compliance	Trial period	ifdelissecurity.com/platforms/ifdelis-halo
Forcepoint	Forcepoint ONE	Security service edge	By request	forcepoint.com/product/forcepoint-one
	Data Loss Prevention	Data security		forcepoint.com/product/dlp-data-loss-prevention
Fortinet	Universal ZTNA	Zero-trust network access	By request	fortinet.com/solutions/enterprise-midsize-business/network-access/application-access

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
Fortra	Alert Logic	Managed detection and response	By request	alertlogic.com
	Core Security	Threat prevention and identity and access management		fortra.com/product-lines/core-security
	Digital Guardian	Enterprise-wide data protection		fortra.com/product-lines/digital-guardian
	Tripwire	Integrity management		tripwire.com
	Vera	Secure ifle collaboration and digital rights management		fortra.com/product-lines/vera#products
Forum Systems	Forum Sentry	API gateway	By request	forumsys.com/product-solutions/forum-sentry
Gigamon	ThreatINSIGHT	Network detection and response	By request	gigamon.com/products/detect-respond/gigamon-threatinsight.html
	Application Filtering Intelligence	Application identiifcation and trafifc forwarding		gigamon.com/products/optimize-trafifc/application-intelligence/application-ifltering-intelligence.html
GitGuardian	Internal Monitoring	Code commit security	Free tier	gitguardian.com/monitor-internal-repositories-for-secrets
	Public monitoring	GitHub secrets monitoring	By request	gitguardian.com/monitor-public-github-for-secrets
GitLab	GitLab	DevOps platform	Free tier	about.gitlab.com
	GitLab	DevSecOps	Trial period	about.gitlab.com
GrammaTech	CodeSentry	Software supply chain security	By request	grammatech.com/codesentry-sca
	CodeSonar	Static code analysis		grammatech.com/products/source-code-analysis
HCL	AppScan	Web application security testing and scanning	Trial period	hcltechsw.com/appscan
	BigFix	Endpoint management		hcltechsw.com/bigifx
Hillstone Networks	Application Delivery Controller	Application delivery optimization	By request	hillstonenet.com/products/application-protection/application-delivery-controller
	W-Series Web Application Firewall	Web server, application, and API security		hillstonenet.com/products/application-protection/waf
HP	Wolf Security	Endpoint security	By request	hp.com/us-en/security/pc-security.html
IBM	IBM Security	Enterprise cybersecurity solutions	Trial period	ibm.com/security
iboss	Zero Trust Edge	Zero-trust data protection and security breach prevention	By request	iboss.com/what-is-zero-trust
Imperva	Web Application Firewall	Application and API protection	Trial period	imperva.com/products/web-application-ifrewall-waf
	DDosS Protection	Application layer security		imperva.com/products/ddos-protection-services
	API Security	Continuous API endpoint security	By request	imperva.com/products/api-security
Infoblox	BloxOne Threat Defense	Network security and SecOps	By request	infoblox.com/products/cybersecurity-ecosystem
in-toto	in-toto	Software supply chain security	Open source	in-toto.io
Invicti	Invicti	Application security testing	By request	invicti.com

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
Juniper Networks	Juniper Secure Edge	Full-stack secure services edge	By request	juniper.net/us/en/products/security/secure-edge.html
	SRX Series Firewalls	Network edge, data center, and cloud app security		juniper.net/us/en/products/security/srx-series.html
	Juniper Advanced Threat Prevention	Threat intelligence hub		juniper.net/us/en/products/security/advanced-threat-prevention.html
	SecIntel	Threat-aware network		juniper.net/us/en/products/security/secintel-threat-intelligence.html
	Juniper Secure Analytics	Security information and event management		juniper.net/us/en/products/security/secure-analytics.html
Kali	Kali Linux	Penetration testing	Open source	kali.org
Keysight	BreakingPoint QuickTest	Performance and security testing	By request	keysight.com/us/en/products/network-security/breakingpoint-quicktest.html
	BreakingPoint Cloud	Microsoft Azure DDoS protection validation		keysight.com/us/en/products/network-security/breakingpoint-cloud.html
	Security Operations Suite	Attack and malicious trafifc simulations	Trial period	keysight.com/us/en/products/network-security/breach-defense.html
Lacework	Polygraph Data Platform	Data-driven cloud security	Trial period	lacework.com/platform
Legit Security	Legit Security	Software supply chain security	By request	legitsecurity.com/software-supply-chain-security-platform
LogRhythm	LogRhythm SIEM	Enterprise cybersecurity	By request	logrhythm.com/products/logrhythm-siem
Lookout	Lookout Security Platform	Threat detection and data protection	By request	lookout.com/products/platform
Lumen	Enhanced Cybersecurity	Cybersecurity and email protection	By request	lumen.com/en-us/security/enhanced-cybersecurity.html
	DDoS & Web Application Security	Multi-vector and mixed application layer defense		lumen.com/en-us/security/ddos-and-web-application.html
	Web Application Firewall	Web application security and acceleration		lumen.com/en-us/security/web-application-ifrewall.html
Malwarebytes	Endpoint Protection	Malware protection and remediation	By request	malwarebytes.com/business/endpoint-protection
	Managed Detection and Response	Threat detection and remediation with monitoring		malwarebytes.com/business/managed-detection-and-response
Mandiant	XDR Platform	Extended detection and response	Trial period	mandiant.com/advantage
McAfee	McAfee Mobile Security	Android and iPhone protection	By request	mcafee.com/en-us/antivirus/mobile.html
	McAfee WebAdvisor FREE	Malware and phishing protection	Free	mcafee.com/en-us/safe-browser/mcafee-webadvisor.html
Mend	Supply Chain Defender	Open-source supply chain security	Trial period	mend.io/mend-supply-chain-defender
	Mend SCA	Open-source software management	By request	mend.io/sca
MetaFlows	Metalfows Security System	Threat detection	By request	metalfows.com
Micro Focus	CyberRes Fortify	Application security	Trial period	microfocus.com/en-us/cyberres/application-security
	CyberRes Voltage	Data privacy and protection	By request	microfocus.com/en-us/cyberres/data-privacy-protection

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
Microsoft	Defender for Endpoint	Multi-platform enterprise endpoint device security	Trial period	microsoft.com/en-us/security/business/endpoint-security/microsoft-defender-endpoint
	Defender for Cloud	Cloud access security broker	By request	microsoft.com/en-us/security/business/siem-and-xdr/microsoft-defender-cloud-apps
Microsoft Azure	Web Application Firewall	Cloud-native web application ifrewall	Free tier	azure.microsoft.com/en-us/products/web-application-ifrewall
NETSCOUT	nGenius Decryption Appliance	SSL and TLS trafifc decryption tools	By request	netscout.com/product/ngenius-decryption-appliance
	Omnis Cyber Intelligence	Network threat detection and response		netscout.com/product/cyber-intelligence
	Arbor Sightline	Network visibility		netscout.com/product/arbor-sightline
	Arbor Cloud	Automated DDoS attack protection		netscout.com/product/arbor-cloud
Neustar Security Services	UltraWAF	Web application cybersecurity	By request	neustarsecurityservices.com/web-application-ifrewall
	UltraDDoS Protect	DDoS attack protection		neustarsecurityservices.com/ddos-protection
NGINX	NGINX App Protect	WAF and DoS for application and API protection	Free tier	nginx.com/products/nginx-app-protect
Nmap	Nmap	Network exploration and security auditing	Open source	nmap.org
NowSecure	NowSecure Platform	Static, dynamic, and interactive mobile application testing	By request	nowsecure.com/products/nowsecure-platform
	NowSecure Workstation	On-prem testing kit		nowsecure.com/products/nowsecure-workstation
NSFOCUS	NSFOCUS	Hybrid enterprise security and service providers	By request	nsfocusglobal.com
	Cloud DDoS Protection Services	DDoS attack protection		nsfocusglobal.com/products/cloud-ddos-protection-service-cloud-dps
Okta	Workforce Identity Cloud	Identity and access management	Trial period	okta.com/workforce-identity
	Customer Identity Cloud	Consumer and SaaS application security		okta.com/customer-identity
Onapsis	Onapsis Platform	Cybersecurity for business-critical apps	By request	onapsis.com/onapsis-platform
OpenText	Network Detection & Response	Network visibility for threat defense	Trial period	opentext.com/products/network-detection-and-response
	EnCase Endpoint Security	Endpoint detection and response	By request	opentext.com/products/encase-endpoint-security
OPSWAT	MetaDefender	Advanced threat prevention	By request	opswat.com/products/metadefender
	MetaAccess	Trust endpoint access to cloud and local networks		opswat.com/products/metaaccess
OWASP Foundation	OWASP	Open-source security projects	Open source	owasp.org

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
Palo Alto Networks	Cortex XDR	Extended detection and response	By request	paloaltonetworks.com/cortex/cortex-xdr
	Prisma Cloud	Cloud-native security		paloaltonetworks.com/prisma/cloud
	CN-Series	Kubernetes network security		paloaltonetworks.com/network-security/cn-series
	Prisma Access	Secure access service edge		paloaltonetworks.com/sase/access
Perforce	Perfecto	Mobile and web app testing	Trial period	perfecto.io
PortSwigger	Burp Suite Enterprise Edition	Dynamic web vulnerability scanner	By request	portswigger.net/burp/enterprise
	Dastardly	Web application security scanning for CI/CD	Free	portswigger.net/burp/dastardly
Pradeo Security	Mobile Threat Defense	Mobility and endpoint security	Trial period	pradeo.com/en-US/mobile-threat-defense
	Mobile App Security Testing	Mobile application security and testing	By request	pradeo.com/en-US/mobile-application-security-testing
Progress Chef	Chef InSpec	App and infrastructure testing framework	Open source	community.chef.io/tools/chef-inspec
Proofpoint	Advanced Threat Protection	Threat detection and prevention	By request	proofpoint.com/us/products/advanced-threat-protection
	Proofpoint Information and Cloud Security	Cloud threat protection, access control, and security monitoring		proofpoint.com/us/products/cloud-security
Qualys	Qualys Cloud Platform	Global IT, compliance, and security posture assessment	Trial period	qualys.com/cloud-platform
	VMDR	Vulnerability management, detection, and response		qualys.com/apps/vulnerability-management-detection-response
	WAS	Web application scanning		qualys.com/apps/web-app-scanning
Quest	SpecterOps BloodHound Enterprise	Identify, quantify, and prioritize attack paths	Trial period	quest.com/products/attack-path-management-software
	KACE Systems Management Appliance	IT systems management	By request	quest.com/products/kace-systems-management-appliance
	One Identity Password Manager	Self-service management and security for end-user passwords	Trial period	oneidentity.com/products/password-manager
Radware	Cloud WAF Service	Web application security protection	Trial period	radware.com/products/cloud-waf-service
	Kubernetes WAF	Scalable web application security solution	By request	radware.com/products/kubernetes-waf
	Threat Intelligence	Preemptive multi-layered DDoS protection		radware.com/products/multi-layered-ddos-protection
	Cloud Native Protector	Public cloud infrastructure protection		radware.com/products/cloud-native-protector
Rapid7	Cloud Risk Complete	Continuous cloud security and compliance	By request	rapid7.com/offers/cloud-security-risk-complete-subscription
	InsightIDR	SIEM, extended detection, and response	Trial period	rapid7.com/products/insightidr
	InsightAppSec	Dynamic application security testing		rapid7.com/products/insightappsec
Rezilion	Rezilion	Software attack surface management	By request	rezilion.com/platform

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
RSA Security	ID Plus	Cloud-based identity and access management	Trial period	rsa.com/products/id-plus
	SecurID	On-premises identity and access management	By request	rsa.com/products/securid
Scribe Security	Scribe Platform	End-to-end software supply chain security	Free tier	scribesecurity.com
Secure Code Warrior	SCW Built Integrations	Security training for developers	By request	securecodewarrior.com/product/integrations
Security Compass	SD Elements	Software threat modeling	By request	securitycompass.com/sdelements
Security Innovation	CMD+CTRL Base Camp	Software application security training	By request	securityinnovation.com/training
Securonix	Next-Generation SIEM	Analytics-based SIEM	By request	securonix.com/products/next-generation-siem
	Open XDR	Extended detection and response		securonix.com/products/extended-detection-response
SentinelOne	Singularity XDR	End-to-end enterprise visibility, protection, and response	By request	sentinelone.com/platform
ServiceNow	Security Operations	SOAR and risk-based vulnerability management	By request	servicenow.com/products/security-operations.html
SiteLock	SiteLock	Website security and monitoring	By request	sitelock.com
SonicWall	Network Security Manager	Firewall management	By request	sonicwall.com/products/management-and-reporting/network-security-manager
	Capture ATP	Cloud-based, multi-engine sandbox for threat detection	Sandbox	sonicwall.com/products/capture-advanced-threat-protection
	Cloud Edge Secure Access	Zero-trust and least privilege security	By request	sonicwall.com/products/cloud-edge-secure-access
Sophos	Managed Detection and Response	Cybersecurity-as-a-Service	Trial period	sophos.com/en-us
SPIFFE	Secure Prroduction Identity Framework for Everyone	Universal identity control plane for distributed systems	Open source	spiffe.io
Splunk	Splunk Enterprise Security	Analytics-driven SIEM for threat detection and response	Trial period	splunk.com/en_us/products/enterprise-security.html
	Splunk SOAR	Security orchestration, automation, and response		splunk.com/en_us/products/splunk-security-orchestration-and-automation.html
StackPath	EdgeSSL	SSL management and performance	By request	stackpath.com/products/edgessl
	SP // Web Application Firewall	WAF		stackpath.com/products/waf
Sucuri	Website Firewall (WAF)	Website protection from hacks and attacks	Trial period	sucuri.net/website-ifrewall
	Website Security Platform	Website malware removal and protection		sucuri.net/website-security-platform
Synopsys	Application Security	Application software security	By request	synopsys.com/software-integrity.html
Tenable	Tenable One	Exposure management platform	By request	tenable.com/products/tenable-one

DZONE'S 2022 APPLICATION SECURITY SOLUTIONS DIRECTORY

Company	Product	Purpose	Availability	Website
Thales	Data Protection Solutions	Data protection and compliance	By request	cpl.thalesgroup.com/data-protection
	Access Management	Identity and access management	Trial period	cpl.thalesgroup.com/access-management
The Update Framework	TUF	Framework for securing software update systems	Open source	theupdateframework.io
Threat Stack	Cloud Security Platform	Cloud-native workload compliance and security access	By request	threatstack.com/cloud-security-platform
Trend Micro	Trend Micro One	Cybersecurity platform	By request	trendmicro.com/en_us/business/products/one-platform.html
UBIKA	UBIKA	Web app and API security	By request	ubikasec.com
UpGuard	UpGuard BreachSight	Attack surface management	Trial period	upguard.com/product/breachsight
	CyberResearch	Managed third-party risk and data leak protection		upguard.com/product/cyberresearch
	UpGuard Vendor Risk	Third-party risk management		upguard.com/product/vendorrisk
Varonis	DatAdvantage Cloud	Cloud security	By request	varonis.com/products/datadvantage-cloud
	DatAdvantage	Identity and access management		varonis.com/products/datadvantage
	DatAlert	Threat detection		varonis.com/products/datalert
	Federal Policy Pack	Data protection		varonis.com/products/federal-policy-pack
Venaif	Venaif Control Plane for Machine Identities	Identity and access management	Trial period	venaif.com/control-plane
Verizon	Network Detection and Response	Network visibility, threat detection, and forensic analysis	By request	verizon.com/business/products/security/managed-detection-response-services/network-detection-response
VMWare	Carbon Black Cloud	Endpoint and workload protection platform	By request	vmware.com/products/carbon-black-cloud.html
	Network Trafifc Analysis	Anomalous activity and malicious behavior detection		vmware.com/products/network-trafifc-analysis.html
	NSX Distributed Firewall	Layer 7 internal ifrewall		vmware.com/products/nsx-distributed-ifrewall.html
	NSX Network Detection & Response	Event correlation across multiple detection engines		vmware.com/products/nsx-network-detection-response.html
Wallarm	API Security Platform	End-to-end API security	By request	wallarm.com/product/wallarm-cloud-native-platform-overview
Waratek	ARMR Platform	Security-as-Code engine	By request	waratek.com/products
Wireshark Foundation	Wireshark	Network protocol analyzer	Open source	wireshark.org
ZScaler	Zscaler Zero Trust Exchange	Security service edge	By request	zscaler.com/platform/zero-trust-exchange