ShiftLeft

ShiftLeft

Economic Value Generated by ShiftLeft's Approach to Modern Application Security

(Code Analysis & Runtime Security)

Table of Contents

OVERVIEW

Abstract *- Today's application security slows down the software development pipeline. Current tools have major operational inefficiencies in the value chain that make app-sec slow and expensive. ShiftLeft's breakthrough innovations throughout the app-sec value chain alleviate massive amounts of inefficiencies and have generated significant economic value for customers.*

In today's hypercompetitive world, technology powered by agility is a great competitive advantage. Amazon, despite its behemoth size, stays more agile than most startups to sustain their competitive advantage. Alibaba-the world's largest online marketplace operator-in 2017, by the sheer dint of software technology and agility, launched and became the world's largest mutual fund in just nine months. Their rapid pace of software delivery took behemoths like Vanguard and JPMC by surprise.

In just the last 10 years, average software release frequency has dropped from months to many times a day. Much of this can be credited to modern CI/CD, efficiencies of hardware virtualization, software-defined networks, cloud infrastructure, serverless, and the containerization movement. Much of the modern software supply chain is driven by the motto of rapid development, automation, componentization, rapid deployment, and feedback insights from production.

However, application security tools have not kept pace with these advancements. Modern code analysis (SAST, DAST, IAST, SCA) or runtime security tools (WAF, RASP) falter on one or more of the following value drivers-leading to inefficiencies in the value chain.

	Value Driver	Why Does It Matter?
1	**Faster speed of analysis and/or protection**	Faster analysis speeds up software delivery; faster delivery → faster time to market → more revenue For runtime tools, it is the cost of human time to construct policies to make WAF/RASP work For static tools, it is the cost of latency caused in the CI/CD pipeline by interventions during code analysis
2	**Low false positives** (misidentification of genuine software interaction as a security issue)	**False positives cause issues up and down the development pipeline** **✓ Time/money lost by diversion of development resources to investigate false positive cases** **✓ Time/revenue lost by slowdown of releases** **✓ Revenue lost by identifying genuine customers interaction as security issue**
3	**Comprehensive analysis**	**Many code analysis tools don't look at 3rd party or open source code. Some only look at 3rd party code. Most do not examine for zero-day vulnerabilities etc. Any such vulnerability that goes undetected causes** **✓ Dev cost in terms of time spent to fix issues later in the cycle (post deploy, post production)** **✓ Operations cost to provide virtual patch in production** **✓ Revenue impact due to security incident triggered through such components**

4	Understanding of app code/semantics *(during runtime protection)*	A runtime security tool that doesn't understand application code/semantics can't understand the app's attack surface, hence, will always be limited in its ability to protect an app. Lack of such knowledge causes ✓ Engineering cost to perpetually write/manage security policies for the application ✓ Performance impact due to excessive instrumentation of app
5	Low performance impact *(during runtime)*	Heavy instrumentation as RASP solution or excessive policies in WAF will cause ✓ Cost of performance engineer to find/fix issues ✓ Customer conversion impact due to adverse user experience ✓ Additional cost of infrastructure to speed up performance

HOW DO CURRENT APPLICATION SECURITY TOOLS STACK UP?

Henceforth, any tool can be judged on whether it creates economic value by reducing cost (or negative revenue impact) of inefficiencies due to the above factors. Following section examines current application security tools on above factors:

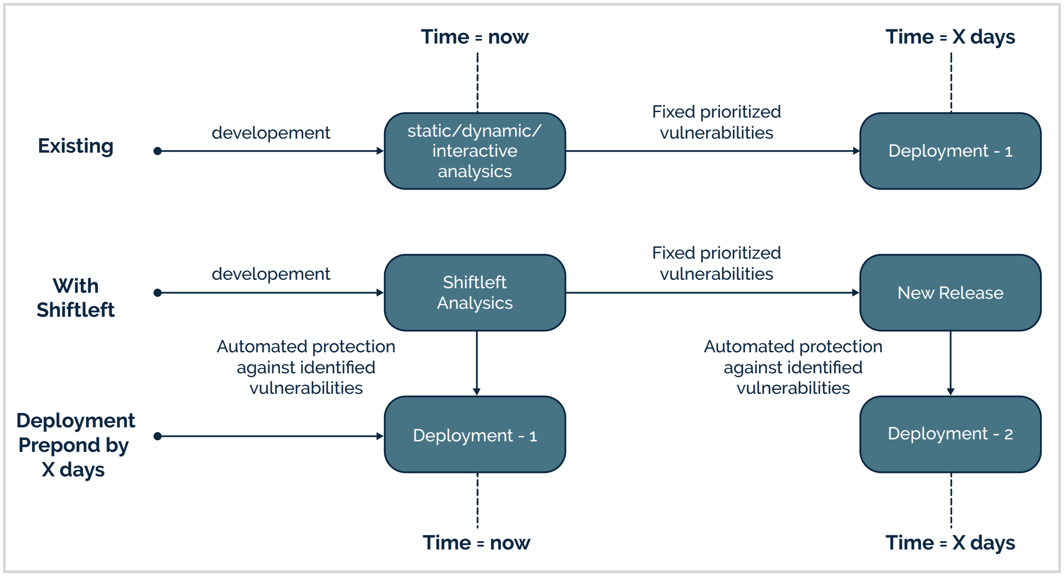
- Traditional SAST by the virtue of large false positive rates requires the cost of at least one developer per medium-size application (to find and investigate FPs).
  - Also with an increasing number of open source components per software (57% as an average), lack of visibility on software components means developers working on investigating vulnerabilities discovered in production.
- DAST catches fewer vulnerabilities (as it is based on request-response), doesn't identify vulnerability source in the code, is not designed to be part of CI/CD automation and consequently requires an engineer to realize full value during development.
- IAST, being an instrumentation-led technology, requires a high cost of integration and maintenance. Based on an estimate, such tests require a build engineer's time to integrate and eventually maintain.
- RASP has been known to suffer from performance impact on the code due to excessive instrumentation of code (as it lacks knowledge of application vulnerabilities/semantics). Apart from performance costs, RASP requires full-time engineers to configure and manage security policies for each application.
- A WAF lacks knowledge of applications and consequently requires 1-2 full-time engineer(s) for policy and deployment management. Policies that are constructed without application knowledge generate a lot of false positives, triggering wasted engineering cycles of investigation. Finally, the need for human mediation causes it to be disconnected with the speed of SDLC.

For the non-tool option, penetration testing is marginally better than DAST. Such simulations are not as exhaustive as a single tool and are labor intensive and hence cannot be automated due to their reliance on the human element. Wrapping all of the above in a tabular format, we find the following:

Value Driver	Does the Tool Deliver? (For Each Specific Value)				
	Code Analysis Tool		Runtime Protection Tool		
	SAST	DAST	IAST	RASP	WAF
Fast speed of code analysis or fast speed of app protection deployment (runtime protection)	No	Yes	Yes, Partially	Yes	No
Low false positives (misidentification of genuine software interaction as a security issue)	No	No	Yes	Yes, (Less Severe)	No
Comprehensive analysis	No	No	Partially	No	No
Understanding of app code/semantics (during runtime protection)	n/a	n/a	n/a	No	No
Low performance impact (during runtime)	n/a	n/a	n/a	No	Yes

HOW DOES SHIFTLEFT RECAST THIS VALUE CHAIN?

ShiftLeft has innovated throughout the app security value chain to reduce massive amounts of existing inefficiencies. Let us go through each value chain innovation one by one.

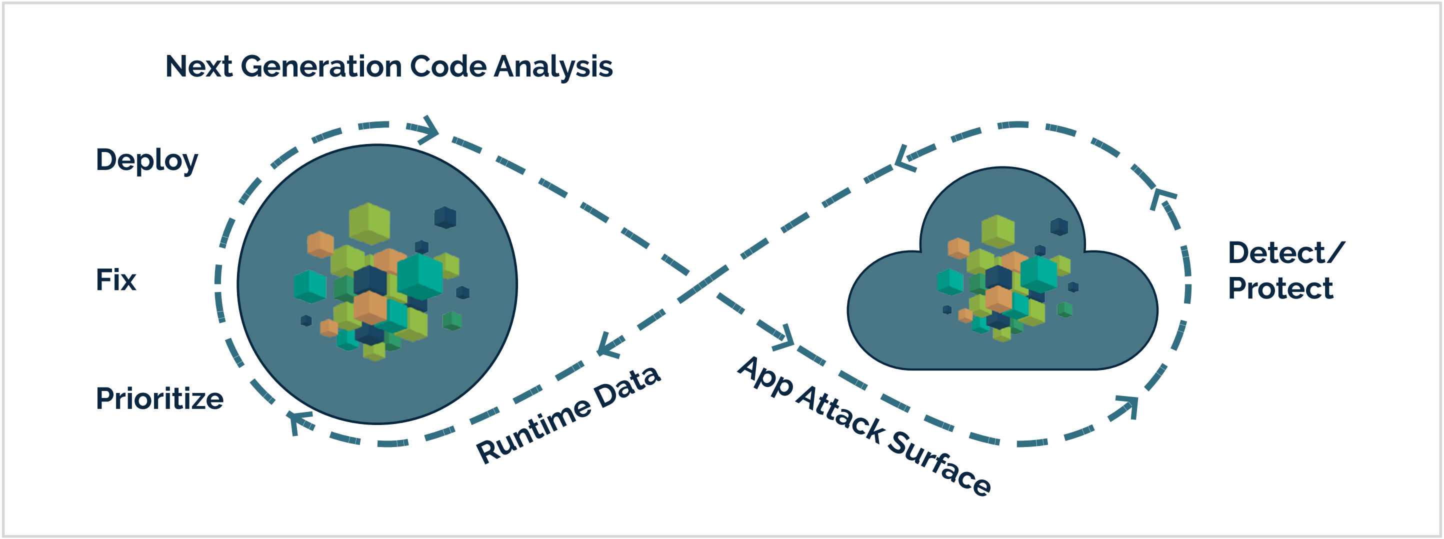


Time = now
Time = X days
Existing
developement
static/dynamic/ interactive analysis
Fixed prioritized vulnerabilities
Deployment - 1
With Shiftleft
developement
Shiftleft Analysis
Fixed prioritized vulnerabilities
New Release
Automated protection against identified vulnerabilities
Automated protection against identified vulnerabilities
Deployment Prepend by X days
Deployment - 1
Deployment - 2
Time = now
Time = X days

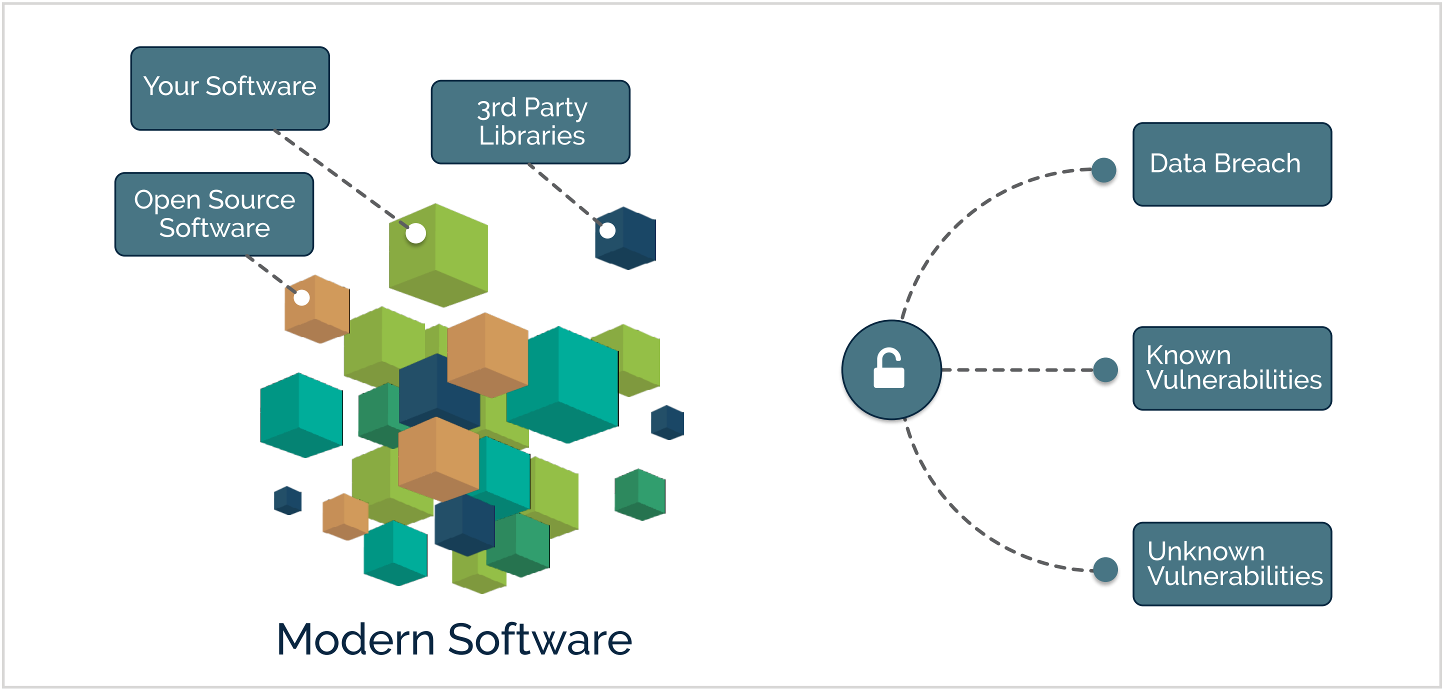
1. Speed Breakthrough - In the traditional application security workflow, developers are expected to fix a certain number of vulnerabilities BEFORE the application can be deployed. In some cases, a WAF is deployed for virtual patches before an application can be approved for deployment.

However, ShiftLeft automatically creates a security layer to prevent against the exploitation of each vulnerability bug discovered during code analysis of the application. It does this in a single build–deploy cycle, eliminating deployment time lag as caused by tools like WAF.

2. False Positive Breakthrough - Unlike traditional application security testing tools, ShiftLeft provides for automated vulnerability validation through its runtime observation or simulated runtime. It automatically signals all the vulnerabilities that are true and hence exercisable. This eliminates the quantum of effort/time that a developer must spend to investigate a vulnerability for it being a false positive.

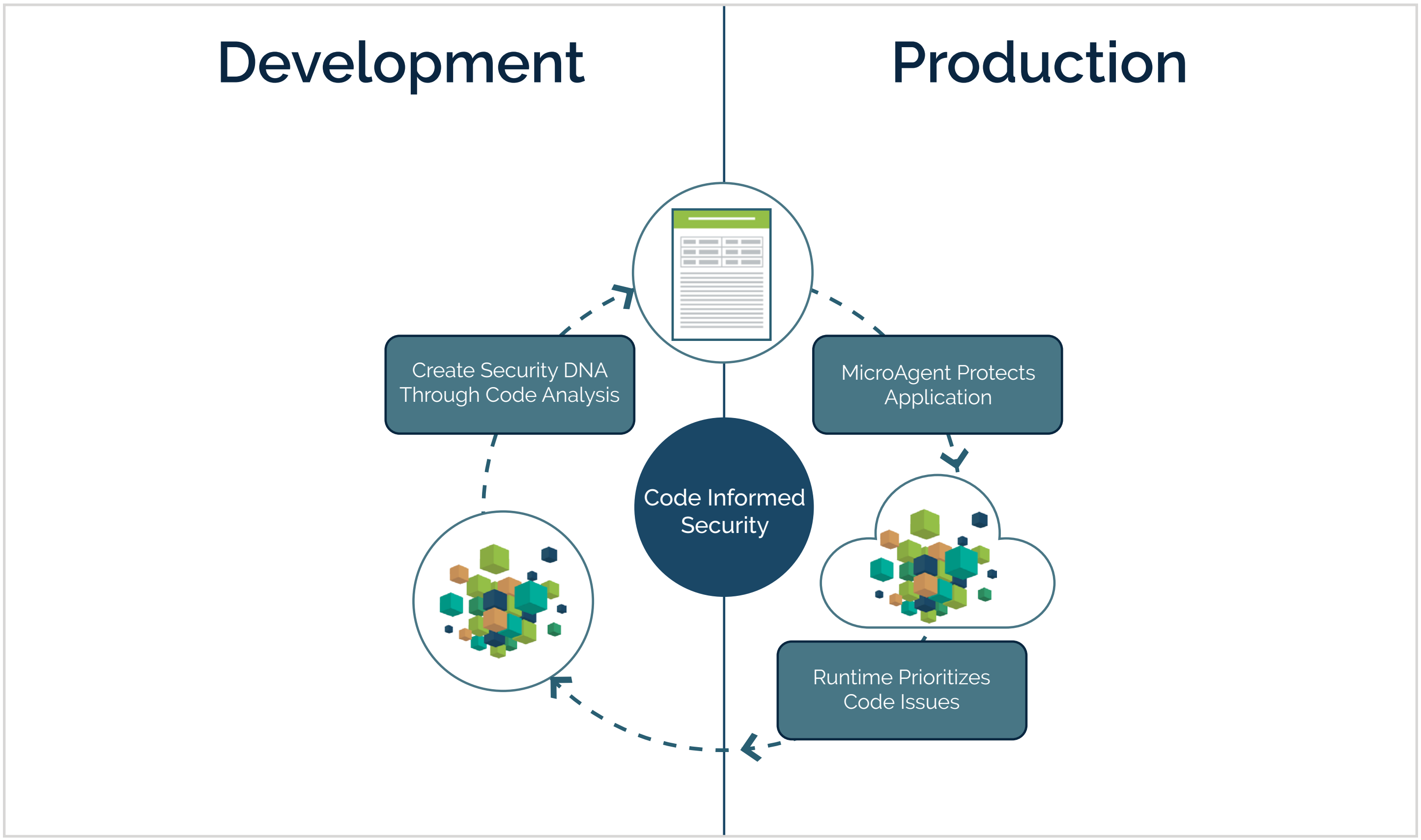


3. Component/3rd Party Analysis Breakthrough - ShiftLeft's innovation allows component analysis, both through static routes and with the use of runtime traffic. Its underlying code property graph technology allows the full application to be analyzed. On the other hand, SAST tools cannot do component analysis by themselves; IAST tools do this analysis based on testing scripts.



4. Code-Informed Runtime Protection Breakthrough - There are exactly ZERO web application firewalls or RASP solutions that can automatically analyze the code of the application to build out security policies to defend an application. At best, WAF admins tend to rely on estimation or results of penetration testing to define security policies for an application.

ShiftLeft's innovative approach uses the results of code analysis to automatically create security policies to defend application during production. This not only facilitates well-directed security policies, but also eliminates the cost or time spent to manually construct such policies.


Development
Production
Create Security DNA Through Code Analysis
MicroAgent Protects Application
Code Informed Security
Runtime Prioritizes Code Issues

5. Performance Breakthrough - Through its innovation of the code property graph, ShiftLeft requires significantly less time/resources to perform code analysis. Additionally, as ShiftLeft instruments code only at locations identified by code analysis, it has significantly less impact on product performance during runtime. On the other hand, RASP tools instrument all possible parts of code leading to major performance drawbacks.

HOW DOES SHIFTLEFT STACK UP AGAINST CURRENT PLAYERS?

Based on the above, ShiftLeft eliminates inefficiencies across the entire value spectrum and generates economic value. The table below stacks up ShiftLeft against current tools in the spectrum.

Value Driver	Does the Tool Deliver? (For Each Specific Value Driver)					
	Code Analysis Tool		Runtime Protection Tool			**Integrated** (Analysis + Runtime)
	SAST	DAST	IAST	RASP	WAF	ShiftLeft
Fast speed of code analysis or fast speed of app protection deployment (runtime protection)	No	Yes	**Yes,** Partially	Yes	No	Yes
Low false positives (misidentification of genuine software interaction as a security issue)	No	No	Yes	**Yes,** (Less Severe)	No	Yes
Comprehensive analysis	No	No	Partially	No	No	Yes
Understanding of app code/semantics (during runtime protection)	n/a	n/a	n/a	No	No	Yes
Low performance impact (during runtime)	n/a	n/a	n/a	No	Yes	**Yes** (Very Low Impact)

[1]HOW DOES THIS VALUE TRANSLATE IN MONETARY TERMS?

ShiftLeft has the least cost of managing security versus any other application security tool that is available in the industry. From a magnitude perspective, it costs less than 20% than any other application security tool. The following section explains the cost advantages for a single application. Assumptions and calculations are explained in the appendix.

[1] This is an example scenario - we encourage readers to plug in their own estimates for their deployments and try this out.

Value Driver	Cost/Revenue Impact Introduced by Tools					
	Code Analysis Tool		Runtime Protection Tool			Integrated (Analysis + Runtime)
	SAST	DAST	IAST	RASP	WAF	ShiftLeft
Fast speed of code analysis or fast speed of app protection deployment (runtime protection)	$10,000	0	$5000	0	$10,000	0
Low false positives (misidentification of genuine software interaction as a security issue)	$4000 - $16000	$4000 - $16000	0	$4160 - $10160	$2160	0
Comprehensive analysis	$12,800	$12,800	$6400	n/a	n/a	0
Understanding of app code/semantics (during runtime protection)	n/a	n/a	n/a	$3888	$3888	0
Low performance impact (during runtime)	n/a	n/a	n/a	$16900	0	$8200
Total cost/revenue implication	$26800 - $38800	$16800 - $28800	$11400	$24948 - $30948	$16048	**$8200**

As is evident, ShiftLeft has the lowest cost of operation against any other application security deployment in the market today.

SHIFTLEFT VALUE ADVANTAGE

A typical application security deployment contains code analysis + runtime/production application security. The following table demonstrates ShiftLeft's total economic value (in costs saved) vs. any realistic combination of security tools (for a single app).

Lowest Cost App Security Deployment	IAST + WAF
Operational Cost of IAST (For a Single App)	$11,400
Operational Cost of WAF (For a Single App)	$16,048
Total Cost of IAST + WAF	**$27,448**

VS

Cost of ShiftLeft App Sec Deployment
(Singular Installation for Analysis and Runtime)

$8200

☆ ShiftLeft's cost of deployment is just 30% of the next best low operational cost app security combination

☆ Standalone, ShiftLeft's deployment operational cost is 50% of WAF, 22% of SAST, 29% of DAST, 70% of IAST, and 27% of the RASP option

[2]APPENDIX

Data, statistics, and costing used to construct the above tables are covered in the footnote.

---

[2] Some baseline statistics that we would use for this purpose are listed below.

1. Median number of vulnerabilities (prioritized) per application - 20 (source) ; Median False positive rate - 25% or higher (source)
2. Assumed hourly cost for a software developer - 80 USD per hour (source)
3. Average time it takes to fix a vulnerability bug - 40 man-hours (source); Average time to investigate a vulnerability bug - 10 man-hours (assumed)
4. Assumed additional number of bugs per application coming from open source - 20% (source)
5. Cost of a data breach - 4 million USD (source)
6. Average cost of a penetration test per application - 20K USD (source - vendor price list)
7. Average cost of a security engineer - 54 USD per hour (source); Assumed average time to resolve a false positive issue by a security engineer - 8 hours
8. Average time taken to create and virtually patch a vulnerability - 10 hours (assumed derived from OWASP virtual patch cheat sheet)
9. Assumed average time taken to tune a WAF per application - 72 hours
10. Average hourly cost of a performance engineer - 50 USD per hour (source)
11. Assumed average time taken to resolve a performance bug - 16 hours
12. Assumed average performance bugs per application - 8
13. Assumed additional cost for resolving a performance issue - 500 USD on AWS per annum (assuming an upgrade from m2.xlarge to c3.large due to performance issues)
14. Assumed conservative estimate for a performance revenue impact per application per month - 10K USD (derived from this)

- *PS: Market opportunity lost to delayed release on app would be a function of the industry and specific business case associated with an app and hence difficult to categorize. For the current case we would assume a figure of 10000 USD across the board.*
- *Assumed revenue impact of latency due to performance issues differs across Industries. Assuming 10K across the board as a conservative estimate.*
- *Partial impact is calculated at 50% of the full impact cost.*

Cost/revenue impact of ShiftLeft

- Total cost and revenue impact - **8200** USD
  - Cost spent on investigating and fixing performance bugs per application - **3200** USD
  - Revenue impact due to customer conversion loss due to performance impact - **5000** USD per month

Cost/revenue impact of SAST

- Total cost and revenue impact - **26800-38800** USD
  - Cost spent on investigating/fixing false positive cases per application - **4000-16000** USD
  - Cost spent on investigating/fixing OSS/third party cases in later cycle - **12800** USD
  - Revenue Impact due to delayed release - **10000** USD

Cost/revenue impact of DAST

- Total cost and revenue impact - **16800-28800** USD
  - Cost spent on investigating/fixing false positive cases per application - **4000-16000** USD
  - Cost spent on investigating/fixing OSS/third party cases in later cycle - **12800** USD

Cost/revenue impact of IAST

- Total cost and revenue impact - **11400** USD
  - Cost spent on investigating/fixing OSS/third party cases in later cycle - **6400** USD
  - Revenue Impact due to delayed release - **5000** USD

Cost/revenue impact of RASP

- Total cost and revenue impact - **24948-30948** USD
  - Cost spent on investigating/fixing false positive cases per application - **2000-8000** USD
  - Cost spent on investigating/fixing false positive cases during runtime per application - **2160** USD
  - Cost spent on creating/defining policies for an application for runtime security - **3888** USD
  - Cost of temporarily fixing performance Impact due to excessive instrumentation - **500** USD
  - Revenue impact due to customer conversion loss due to performance impact - **10000** USD per month
  - Cost spent on investigating and fixing performance bugs per application - **6400** USD

Cost/revenue impact of WAF

- Total cost and revenue impact - **16048** USD
  - Cost spent on investigating/fixing false positive cases during runtime per application - **2160** USD
  - Cost spent on creating/defining policies for an application for runtime security - **3888** USD
  - Revenue Impact due to delayed release - **10000** US