SYNOPSYS®

Automate AppSec Triage With Machine Learning

Introduction

One of the most significant problems facing application security (AppSec) teams is the amount of time it takes to manage the results returned from automated testing tools. Tests may return thousands of potential vulnerabilities, but most AppSec professionals know that only a small fraction of them are worth the time and effort to remediate. AppSec teams comb through these results and triage them—flagging the ones that should be fixed and weeding out the false positives. This process is extraordinarily time-consuming, repetitive, and tedious—but necessary.

Machine learning offers a solution to this problem. This eBook explores how machine learning can be applied to automate the triage process, and examines solutions already on the market.

Machine learning can be applied to automate the triage process

Applying machine learning to AppSec triage

Every industry and market sector is using machine learning, and it's had a positive impact across the board. For example, the manufacturing industry uses machine learning in predictive maintenance to help foresee and avoid costly equipment failures. In healthcare, hospital emergency departments use machine learning to more accurately predict critical care and hospitalization outcomes. The U.S. military is using machine learning to identify, analyze, and respond to threats much faster than a human can.

Application security is also finding uses for machine learning. Microsoft used machine learning to build internal tools that accurately identify critical, high-priority security bugs, 97% of the time.

Code Dx by Synopsys has taken the vital step of applying machine learning to the AppSec triage process. We hypothesized that by watching how security analysts sort, prioritize, and remediate vulnerabilities from a variety of security testing tools, machine learning could greatly reduce triage time, significantly increase the utility and value of security testing, and free analysts to focus on higher-order security tasks. The data indicates this hypothesis is correct.

Finding what's important

Security testing tools like static application security testing (SAST) and dynamic application security testing (DAST) look for coding errors that can result in security vulnerabilities. They typically run hundreds or thousands of rules designed for a unique use case. Some identify security vulnerabilities like cross-site scripting and SQL injection. Others find style issues, such as how comments are worded.

The results—or findings—of scans from a single testing tool can easily number in the thousands. Triage is the process of reviewing and categorizing the findings from a scanning tool or penetration test to eliminate false or unimportant findings, and determining and prioritizing remediation steps. The triage process typically groups findings into three categories:

- **Exploitable true positives.** These are vulnerabilities that exist within the code and that can be accessed by an attacker. This is a narrow definition because there are likely to be countless vulnerabilities that exist in the code but that can't be exploited because there is no way to execute them without access to the code itself. The vulnerabilities identified as exploitable true positives could be used to gain a foothold on a device, run malicious code, affect performance, or cause other unwanted activities. Security and development teams treat these as high-priority findings that must be remediated quickly.
- **Insignificant.** The goal of security is to achieve acceptable risk, not zero risk. Unlike exploitable true positives, findings categorized as insignificant include true positives that can't be reached by an attacker, or those of extremely low criticality, so they won't be scheduled for remediation. This category also includes informational findings typically unrelated to security (such as style rules) that are considered "noise" by most teams.
- **False positives.** These are findings where the tool has mistakenly identified secure code as vulnerable.

From a security viewpoint, the first category is the most important. The goal of the triage process is to identify the most important security issues to remediate, and suppress all others.

Noise in findings leads to longer triage

Finding exploitable true positive issues is critical to producing secure software. However, security testing tools like SAST are notorious for generating false and extraneous findings. A recent study of SAST tools from National Institute of Standards and Technology (NIST) shows that most of the findings are false positives or noise. As shown in the table on page 4, less than a third of findings for the C programming language were true positives. The rest of the findings—68% of the total—were either false positives or insignificant. The results for PHP and Java were slightly better, but still included an effective false positive rate of 50% and 40% respectively. From a triaging standpoint, 68% of the triaging exercise in C language is wasted effort, unnecessary time spent that could have been avoided if the tools were more precise. For Java, "only" 40% of the time is wasted.

The economics of triage

Reducing the amount of trivial and false positives in findings is important because triage is an expensive process. Research by Grammatech (and supported by other studies) found that it takes an analyst 10 minutes to triage a single finding. In a typical deep scan, a single tool can return thousands of findings, and best practice would be to use multiple tools, each producing a combination of unique and redundant findings. As shown in the following table, even moderate results of 1,000 findings can take over 20 days to triage. In today's DevOps environments with multiple releases per day, teams would quickly fall behind by dozens of releases.

Applying this to the results from the NIST research clearly demonstrates the cost of inaccurate and noisy tool output. Had the tools been more accurate and precise (or if the triage process were automated), organizations could save between 8.3 and 14.2 days for every 1,000 findings generated by testing tools. Put another way, for every 240 insignificant or false findings eliminated, an organization saves a full work week of effort by a security analyst.

COST OF TRIAGE			
Number of findings	1,000	5,000	10,000
Triage time per finding	10 min.	10 min.	10 min.
Total triage time (days)	20.8	104.2	208.3
Days triage of false/insignificant findings: C	14.2	70.8	141.7
Days triage of false/insignificant findings: PHP	10.4	52.1	104.2
Days triage of false/insignificant findings: Java	8.3	41.7	83.3

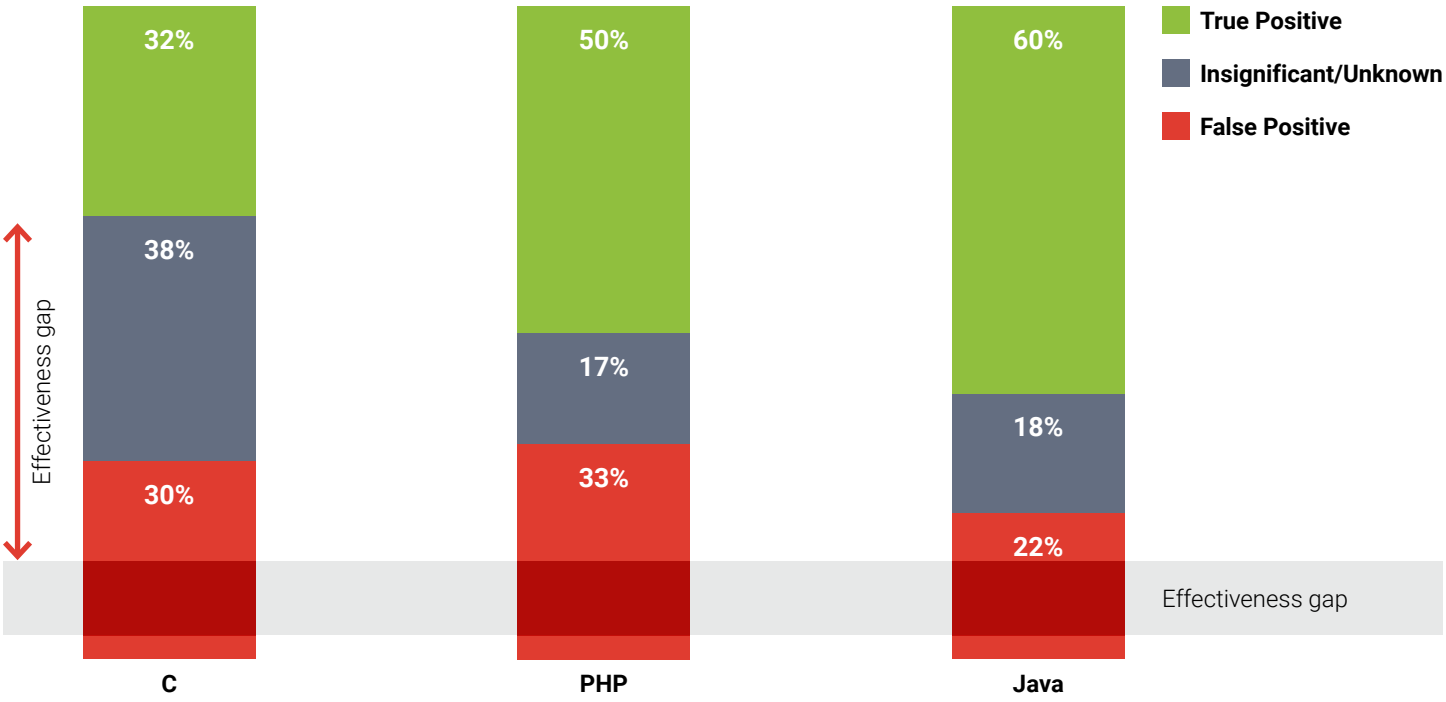
Poor accuracy leads to poor adoption

Identifying exploitable vulnerabilities is important, and adopting SAST and DAST tools are proven ways of doing so. At the same time, development teams are dealing with constantly shortening deadlines for delivering new functionality. Even moderate levels of issues, false positives, and insignificant results that don't warrant remediation can prevent developers from adopting these testing tools.

Microsoft has a very strong software security culture. It surveyed 2,000 of its software engineers in 2016 to learn what constituted the highest acceptable false positive rate for security testing tools. A full 90% of the developers stated that a 5% false positive rate would be acceptable. However, that number dropped quickly as the theoretical false positive rate increased. Only 46% would accept a 15% false positive rate, and only 24% would accept a 20% false positive rate. A similar study by Google found that developers required tools that produced less than 10% effective false positives. Note that Google developers consider an issue to be an effective false positive if the developer elects to not act on the issue.

The effectiveness gap

Given that the NIST study shows effective false positive rates of 40% to 68%, a tremendous effectiveness gap exists between the raw output of the studied tools and the requirements of developers. The table below represents the reported acceptable false positive rates.



Closing the effectiveness gap with machine learning

Automating triage is critical to lowering the financial costs and barriers to adoption caused by noisy and inaccurate findings. Code Dx is applying new machine learning technology to solve the problem. Code Dx's Triage Assistant uses the results from an organization's earlier triaging process to learn what's important to the specific development and security team for each application. It consumes all results and accounts for those that were prioritized and those that were suppressed. Triage Assistant then uses that knowledge on future scans to predict which findings should be acted on and which can be safely ignored, eliminating manual triage.

Different organizations have different priorities when it comes to application security. Some will want to focus immediately on eliminating SQL injections, while others will first target cross-site scripting. Triage Assistant learns from each individual organization's best practices and adopts them for future scans. There's no one-size-fits-all solution that will work for every application and organization. Therefore, the only path forward is to provide a framework that customizes a solution for each individual set of circumstances. Code Dx's Triage Assistant learns what's important to you for each application, and triages subsequent findings accordingly.

Real-world results

Validating automated triage requires organizations to consider the accuracy of machine-made decisions. This includes:

- **Precision.** Misclassifying a true positive as a false positive would leave applications vulnerable. The validation process measures how accurately Triage Assistant predicts false positives or insignificant findings, and expresses the result as the percentage of findings correctly predicted. For example, a result of 100% would mean the model accurately predicted all false positives and did not misidentify any true positives as false positives.
- **Recall.** Failing to identify a false positive or insignificant finding means a human must manually triage that item. The Recall metric measures the percentage of findings that the model correctly predicted to be false positive or insignificant.
- **Accuracy.** The validation process also identifies the percentage of findings that the model correctly predicted to be relevant or irrelevant.

Code Dx worked with five customers and over 1 million triaged findings to validate Triage Assistant. A portion of the previously triaged findings were used to train Triage Assistant. The remaining were analyzed, and the results were compared to the known "ground truth" to measure the accuracy of the model.

RESULTS			
Customer	**Precision**	**Recall**	**Accuracy**
Code Dx Internal	99.87%	99.84%	99.94%
Customer 1	100.00%	99.99%	99.99%
Customer 2	96.48%	96.12%	96.24%
Customer 3	99.99%	99.98%	99.99%
Customer 4	99.22%	98.94%	99.97%
Customer 5	99.81%	99.85%	99.71%

Overall, Code Dx's Triage Assistant was 99.31% accurate in the first pass on this data set. Over time, the model continues to learn and improve based on decisions made by an organization's security analysts.

Applying this result to the NIST research data provides a theoretical savings that could be achieved by using Triage Assistant to eliminate false positives and insignificant findings from the total set of issues requiring review. The potential time savings is significant—the **minimum** amount of time saved is 8.3 days, while the **maximum** amount of labor eliminated is a staggering **140.7 days**, just by using Triage Assistant.

Code Dx is smart automation

The machine learning capability in Triage Assistant is just one example of the smart automation that Code Dx delivers to development, security, and operations teams. The Code Dx platform automates the resource-intensive workflows needed to find, analyze, and fix software vulnerabilities—at DevOps speed. Application security analysts use Code Dx smart automation to help orchestrate tests, prioritize vulnerabilities, track remediation, and continuously assess security risks across the software life cycle.

Code Dx uses machine learning to optimize results from application security testing tools, and more efficiently use and focus the valuable time of AppSec analysts. Code Dx's Triage Assistant minimizes triage time, helping organizations maintain a fast-paced DevOps schedule.

SYNOPSYS®

The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations optimize security and quality in DevSecOps and throughout the software development life cycle.

For more information, go to www.synopsys.com/software.

Synopsys, Inc.
690 E Middlefield Road
Mountain View, CA 94043 USA

Contact us:
U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com