Measuring and Comparing Static Analysis Tools for Web Security

MIGUEL DIDEO and KYLE PEIMAN

ACM Reference Format:
Miguel DiDeo and Kyle Peiman. 2022. Measuring and Comparing Static Analysis Tools for Web Security. 1, 1 (December 2022), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROBLEM STATEMENT

Static analysis tools are very commonly used to analyze code to identify any security vulnerabilities that could be exploited by adversaries. However, not all static analysis tools are created equally, and thus there needs to be a methodology to evaluate what static analysis tools are the best for specific circumstances. We look to use the OWASP Benchmark to analyze a series of static analysis tools to determine which tools are best for which jobs.

2 RELATED WORK

Related Work has been done by Software Assurance Metrics and Tool Evaluation (SAMATE) from NIST, by OWASP's Benchmark for Securtity Automation (BSA), and in Bench marking Static Analysis Tools for Web Security by Paulo Nunes et.al [2018], a paper that was published on the IEEE website and analyzes bench marking tools and looks to improve them. NIST(National Institute of Standards and Technology) and OWASP(Open Web Application Security Project) are two well known organizations that look to make application security easily accessible to everyone either via community projects or research.

OWASP's BSA [1] is relatively user friendly and generates fast and intuitive results. BSA will run a tool over many different test cases that OWASP has created and detect the amount of true positives and false positives that a static analysis tool will pick up on. BSA then generates a score for the tool based on these results. Unfortunately, BSA only comes with 5 tools built into it, and using a tool other than the built in ones is a difficult and involved task. OWASP is best known for their top 10 list that they release every few years that goes over the top 10 most critical security risks.

SAMATE [2] is a large database of tools generated by NIST. SAMATE looks to assist users in picking which tool is the proper tool for their particular use case. They hold the Static Analysis Tool Exposition (SATE), where tool makers can bring their tools and run them against some test cases. The results are then shared at a workshop, and a report is released to the public later on.

Authors' address: Miguel DiDeo; Kyle Peiman.

XXXX-XXXX/2022/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The work done in Benchmarking Static Analysis Tools For Web Security, an academic paper written by Paulo Nunes, et.al [2018] relates to the work we look to do in our project. The team on this project found the benchmarks by OWASP and NIST lacking and sought to create their own benchmark specifically for static analysis tools (SATs) for web security. The team uses five different static analysis tools; phpSAFE, RIPS, WAP, WeVerca, and Pixy. To benchmark these tools they took 143 WordPress plugins from the WPScan Vulnerability Database and analyzed the amount of true positives and false positives received from their analysis tools, they did this for both SQL injection (SQLi) and cross site scripting (XSS). They also separated the files they analyzed by quality, since lower quality plugins will have more vulnerabilities they found it useful to make this divide when ranking the SATs [3].

There are many different ways in which this team could have bench marked their tools, they chose to use the SAMATE methodology and OWASP's benchmark. SAMATE metrics are Precision, F-Score, Recall and Discrimination Rate, however, Discrimination rate involves comparing a vulnerable plugin to a patched version of the vulnerable plugin, which for most vulnerable plugins does not exist, so they decided to not use Discrimination rate. BSA is Benchmark Accuracy Score, it is calculated as $BSA = (TPR - FPR) * 100$, where TPR is the true positive rate and FPR is the false positive rate.

The main contribution this project proposes is their concept of a "workload", that being, the criticality of the application being tested (i.e, business-critical to lower-quality applications). This criticality acts like a kind of weight, every missed bug in the highest critical applications could be disastrous, whereas every false positive in the lowest critical category is a waste of resources that could be spent elsewhere. They wanted to find a tool that would catch all of the vulnerabilities in the most critical software, regardless of how many false positives there were, and the least amount amount of false positives possible for the lowest critical software, while still catching some potential risks.

3 APPROACH

We will take a similar approach to the work of Paulo Nunes et.al. by evaluating the following series of Static Analysis Tools with the OWASP Benchmark:

- FBwFindSecBugs (v1.4.0, v1.4.3, v1.4.4, v1.4.5, and v1.4.6)
- FindBugs (v3.0.1)
- OWASP ZAP (vD-2015-08-24 and vD-2016-09-05)
- PMD (v5.2.3)
- SonarQube Java Plugin (v3.14)
- VisualCodeGrepper (v2.2.0)
- ShiftLeft CORE's NG SAST

We will compile the results for each tool and then begin our analysis considering true positive rate (TPR), false positive rate (FPR),

and using the BSA score for each tool. Speed is also an important metric to consider when comparing different tools. If a tool takes too long to complete its analysis then the tool is also not very usable.

For each tool, all vulnerability categories that are tested in the OWASP Benchmark will be considered when evaluating tools. This includes command injection, insecure cookies, LDAP injection, path traversals, SQLi, trust boundaries, weak encryption algorithms, weak hashing algorithms, weak randomness, XPath injection, and XSS. Alongside the previously mentioned categories, the benchmark also calculates the run time for each tool which will also be taken into consideration when evaluating static analysis tools.

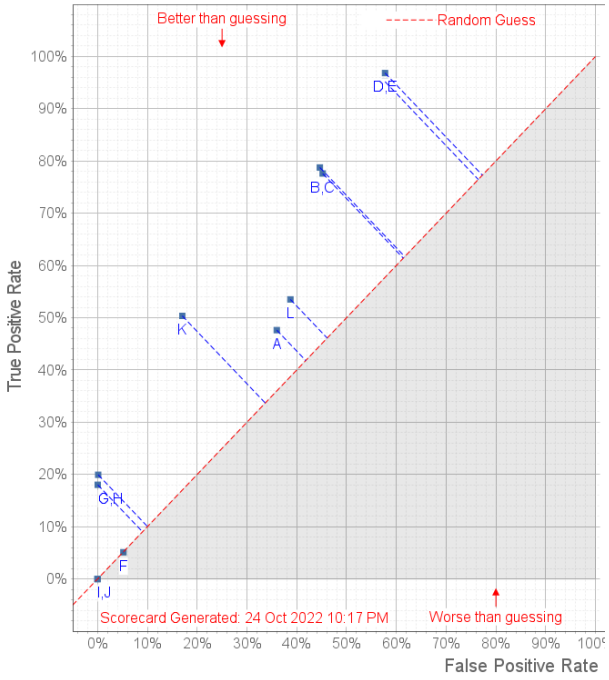
4 EVALUATION

Most of the tables found in Appendix A (i.e., every table but Table 1), show each tool being measured for each of the tested vulnerability category in terms of its TPR, FPR and BSA score. Additionally, each tool has an overall total for TPR, FPR, and BSA score calculated by averaging the total scores in each category divided by the number of categories tested.

The figures found in Appendix B, show a visual representation of the scores shown in the aforementioned tables. Each figure (excluding Figure 1) charts each TPR, FPR, and to an extent BSA score for every category, with larger BSA scores represented by larger distances from the average red line and smaller scores being closer to the red line. The aforementioned red line represents the hypothetical results if each test case evaluated was randomly guessed as a true positive or a false positive. The overall averages for each static analysis tool are also plotted, represented by a large pink dot.

Figure 1, acts as a guide for how to read these scorecard results. Through this, we can see the ideal vulnerability detection zone is in the top left corner, with high TPR values and low FPR values. Having a high TPR and high FPR correlates with a tool that reports almost everything to be vulnerable. Any tool where the TPR and FPR both fall around 50.00% seems to report on vulnerabilities at random. Tools that have both a low TPR and FPR report that nearly nothing is vulnerable, or do not work for that specific vulnerability category. Finally, any time the BSA score is less than 0.00%, the tool performs worse than it would have if it was guessing for every test case.

To showcase how the scorecards work, let's take a look at the combination of all the overall BSA scores for the non-commercial SATs (everything but ShiftLeft Core's NG SAST).



Here, we can see the random guess line split the graph into two halves. The top half is considered better than guessing whereas the bottom half is considered worse than guessing. Higher TPR values are indicated by being further up on the vertical axis and higher FPR values are indicated by being further out on the horizontal axis. Higher BSA scores are shown by being further away from the random guess line. The good news is that most of the tools tested have their overall BSA scores at or above the random guess line, with those dipping below the line only dipping slightly below it. The breakdown of how each overall BSA score is determined will be better shown when each tool's results are broken down in-depth in the next section.

Table 1 in Appendix A covers the run times for each tool. Overall no tool ran over 5 minutes and 30 seconds, which is decent considering the amount of test cases covered. PMD technically ran the fastest in just 11 seconds, but considering that PMD doesn't detect any of the vulnerabilities tested for we wont count its result. FBwFindSecBugs actually ran the fastest, running in 1 minute and 50 seconds in total, just barely edging out FBwFindSecBugs v1.4.3 that ran in 1 minute and 58 seconds. SonarQube Java Plugin v3.14 ran the slowest in 5 minutes and 30 seconds just behind OWASP ZAP vD-2015-08-24 that ran in 5 minutes flat. OWASP ZAP vD-2016-09-05, ShiftLeft Core's NG SAST and VisualCodeGrepper v2.2.0 have Time not specified for their results, so they may fall somewhere in between those results.

5 DISCUSSION

After running the OWASP Benchmark for each tool, each tool will be elaborated upon with details about our findings for each tool

listed separately in the following subsections.

5.1 FBwFindSecBugs

5.1.1 FBwFindSecBugs v1.4.0. Using the data found in Table 6, from Appendix A, we can see that FBwFindSecBugs v1.4.0 has the third worst overall BSA score of any tool tested at 11.65%. However, this comes with a caveat that the overall TPR is the seventh highest at 47.64%, which is higher than some tools with larger overall BSA scores, such as SonarQube Java Plugin v3.14, and both OWASP ZAP versions tested. The best performing category for FBwFindSecBugs v1.4.0 is Weak Random Number with a BSA score of 100.00%, which is the only category where this tool gets a perfect score. Insecure Cookies and Trust Boundary Violations go undetected by this tool, and XPath Injections seem to be always reported as vulnerable with a TPR and FPR of 100.00%.

Where FBwFindSecBugs struggles is with Command Injections, LDAP Injections, and SQLi, which all have FPR's exceeding TPR's, resulting in negative BSA scores. The least negative of these scores is SQLi, with a BSA score of -1.93% with the BSA score for Command Injections ibeing slightly worse at -2.19%. But the worst by far is for LDAP injections, with a BSA score of -13.31%, which is the worst BSA score for any one category in the entire set of data. This overall average performance is also shown in the scorecard for this tool, displayed in Figure 2 with almost all of the scores falling close to the random guess line, with the only decent performances being Weak Hashing Algorithms, the aforementioned Weak Random Number, and Weak Encryption Algorithms, which itself is marred by a decently high FPR.

5.1.2 FBwFindSecBugs v1.4.3. Using the data found in Table 3, from Appendix A, we can see that there is an improvement in the overall BSA score, moving up to 32.39% which is the sixth highest of any tool tested. This is shown with a massive improvement in overall TPR which is now at 77.60%, but hampered by an also increasing overall FPR of 45.21%. Compared to the previous version, there are also improvements in most vulnerability categories. Insecure cookies now have a BSA score of 100.00% and the scores for other already decent categories got better such as Weak Hashing Algorithms, Weak Random Number, and Weak Encryption Algorithms.

Additionally, the three categories in the negatives are now in the positives, with the overall BSA scores for Command Injection, LDAP Injection, and SQLi now sitting at 9.60%, 9.38%, and 8.62% respectively. The one downgrade in the newer version of FBwFindSecBugs comes with XSS, which was not a strong suit for v1.4.0 where it had a BSA score of 1.22%. This score actually gets worse with v1.4.3 where the BSA score now sits at 0.41%. This trend of higher TPRs and FPRs is also shown in the scorecard for this tool, displayed in Figure 3 as while scores are overall higher, more points trend more towards the zone where everything is reported as a vulnerability, best shown with Command Injection, LDAP Injection, SQLi and XPath Injection.

5.1.3 FBwFindSecBugs v1.4.4. Using the data found in Table 4, from Appendix A, we again see general BSA score improvements across the board. The overall BSA score has slightly increased to 34.13%, the fourth highest of any tested tool. This gradual increase also comes with a slightly higher overall TPR of 78.77% and a slightly lower overall FPR of 44.64%. However, the worrying trend of more categories reaching the "reporting everything" zone continues, as shown in Figure 4, with Path Traversal now entering the same zone as the previously mentioned four Injection categories (Command, LDAP, SQL, and XPath) as while some of the FPRs for those categories did go down, it was not nearly enough to escape the top right corner of the chart.

5.1.4 FBwFindSecBugs v1.4.5. Using the data found in Table 5, from Appendix A, we can see that the overall TPR has increased to 95.20% and the overall FPR has increased to 57.74%, giving us an overall BSA score of 37.46%, now the third highest overall BSA score of any tool tested. Trust Boundary Violations are now being detected, though with a BSA score of 0.53%, and TPRs and FPRs in the low 80.00% range guesses are not exactly accurate. However, the bigger jump comes with XSS, which now has a 100.00% TPR and a 62.68% FPR for a BSA score of 37.32%, putting it closer to the top-right corner seen in Figure 5, but not quite close enough to be considered reporting everything yet. Finally, as seen in Table 1, run time has made a significant improvement from v1.4.4

5.1.5 FBwFindSecBugs v1.4.6. Using the data found in Table 6, from Appendix A, the most recent version of FBwFindSecBugs tested with the OWASP Benchmark. In terms of overall numbers, v1.4.6 is a distinct upgrade over v1.4.5 with a higher overall TPR of 96.84% while retaining the same FPR as v1.4.5 for a new overall BSA score of 39.10%, the second highest of any tool tested, and the highest of any non-commercial tool tested.

The slight improvement comes in the form of the TPR for Trust Boundary Violations increasing to 100.00% while retaining a FPR of 81.40%, giving a new BSA score for Trust Boundary Violations of 18.60%. While that does mean it enters the "reporting everything" zone shown in Figure 6, it has the lowest FPR of any category in that grouping. The run time, as seen in Table 1, also continues to be good for v1.4.6, continuing the approach back towards the faster run times of previous FBwFindSecBugs versions. Combining that run time with some of the best performance in the group makes this tool a very good overall choice.

5.2 FindBugs

From the best performing non-commercial tool, we move on to FindBugs v3.0.1, whose results are found in Table 7 in Appendix A, and Figure 7 in Appendix B. FindBugs v3.0.1 is notable for being the worst performing static analysis tool of the bunch, only performing above the random guess line for Path Traversal with a BSA score of 0.77% and performing worse than guessing for SQLi with a BSA score of -1.93%, though with a TPR of 53.68% and a FPR of 55.60%

meaning that SQLi was being detected, but whether or not it was accurately detected was worse than a random guess. All other vulnerability categories were not detected by the tool with TPRs, FPRs, and BSA scores of 0.00%. As a result, FindBugs v3.0.1 was the only static analysis tool tested with an overall BSA score less than 0.00%, with an overall TPR of 5.12%, and an overall FPR of 5.19% giving us an overall BSA score of -0.07%, the worst overall BSA score of any tool tested. As a result, while the run time may be the second lowest of any tool tested according to Table 1, that does not mean the tool should be used for just about anything.

5.3 OWASP ZAP

5.3.1 OWASP ZAP vD-2015-08-24. Using the results found in Table 8 in Appendix A, and Figure 8 in Appendix B. Beginning with the overall results, vD-2015-08-24 has an overall TPR of 18.03% and an overall FPR of 0.04% for an overall BSA of 17.99%, giving us the fifth worst overall BSA score of any tested tool. Most of the reason for this is that while FPRs are low, which is a very good thing, TPRs are not all that high. Additionally, many categories are simply undetected by the tool, including LDAP Injection, Path Traversal, Trust Boundary Violations, Weak Encryption Algorithms, Weak Hashing Algorithms, Weak Random Numbers, and XPath Injections.

Meanwhile for the categories that are detected, Insecure Cookie gets a perfect 100.00% BSA score, but the Command Injection BSA score of 34.92% is the next highest BSA score for any category. SQLi has a BSA score of 34.13% and XSS has a BSA score of 28.86%, with those categories being the only detected categories by the benchmark. This is all reflected in Figure 8, which shows that yes the FPRs are good, but TPRs besides Insecure Cookie are nowhere near the ideal zone. Additionally, the overall run time of this tool, seen in the bottom left corner of Figure 8 and in Table 1 is the largest amount of time reported for any tool in the selection, which could prove to be a downside to some people.

5.3.2 OWASP ZAP vD-2016-09-05. As seen in the results found in Table 9 in Appendix A, and Figure 9 in Appendix B, while there still are not many categories detected, BSA scores across the board in all detected categories have improved with little or no movement in FPR values. This gives the 2016 version an overall BSA score of 19.84% which while it still gives it the sixth worst overall BSA score of any tested tool, it is still an improvement. In terms of the detected categories, it is a bit of a mixed bag in terms of improvements and setbacks with the newer version as SQLi improved to a BSA score of 56.80%, however, XSS remained constant at 28.86% and Command Injection got worse with a BSA score of 32.54% compared to the 34.92% of the previous version. Additionally, the run time, as seen in Table 1 could continue to be an issue as this was one of three tools that did not specify a tool run time.

5.4 PMD

Next is PMD v5.2.3, whose results are found in Table 10 in Appendix A, and Figure 10 in Appendix B. PMD v5.2.3 is notable for not detecting any of the vulnerability categories being tested for, with TPRs, FPRs and BSA scores of 0.00% across the board. This allows for PMD to serve as a control group to compare other tools to. As a result, it also should not be surprising that PMD's run time is the fastest of any tool tested by a long shot, as seen in Table 1. Though, all of that does not mean that PMD should be recommended by any means.

5.5 SonarQube Java Plugin

We now move on to SonarQube Java Plugin v3.14, whose results are found in Table 11 in Appendix A, and Figure 11 in Appendix B. As best shown in Figure 11, the results for the SonarQube Java Plugin really have a little bit of everything. To begin with, the tool has undetected categories such as Path Traversal, SQLi, Trust Boundary, XPath Injection, and XSS. Next, there is a negative BSA score in Command Injection of -2.28%, one of two categories with a high TPR and FPR, the other being LDAP Injection where it is fit squarely in the "flag everything" range with a TPR and FPR of 100.00%.

But despite all of that, everything else is great. Insecure Cookies, Weak Encryption Algorithms, and Weak Random Numbers all get a BSA score of 100.00% and Weak Hashing Algorithm has a BSA score of 68.99% with a FPR of 0.00%. But there is one more caveat to using this tool and that is the run time, clocking in as the second slowest reported tool tested, as seen in Table 1. All of this comes together for an overall TPR of 50.36% and an overall FPR of 17.02%, for an overall BSA score of 33.34%, which is the fifth best overall BSA score of any tool tested, and the perfect definition of a tool that works for specific things, but is nigh on hopeless for anything else.

5.6 VisualCodeGrepper

Next up is VisualCodeGrepper v2.2.0, whose results are found in Table 12 in Appendix A, and Figure 12 in Appendix B. While VisualCodeGrepper does detect almost all of the categories, only not detecting LDAP Injection, Weak Hashing Algorithms, and XSS, what it does detect is not the best with no one category truly entering the top-left corner of Figure 12, though Weak Random Numbers and Weak Encryption Algorithms do get close. Moving over to the top-right of Figure 12, Insecure Cookies and XPath Injections are squarely in the "report everything" zone and Path Traversal is also in that zone but to a lesser extent.

As for the rest, Trust Boundary Violations sit comfortably over the random guess line, but with a TPR of 32.53% and a FPR of 18.60% it's not entirely ideal. However, it's certainly more ideal than the two categories with negative BSA scores, with Command Injection having a BSA score of -1.16% with TPR and FPR in the mid 40% range, and SQLi having an even worse BSA score of -2.54% with TPR and FPR in the low-mid 50% range. This gives VisualCodeGrepper an overall BSA score of 14.78%, the fourth worst of any tool tested. Combine all of that with a not specified run time according to Table 1, and VisualCodeGrepper is hard to recommend, filling a

niche as a jack of all trades, but a master of none.

5.7 ShiftLeft CORE's NG SAST

Last and most certainly not least is the only commercial tool of the bunch: ShiftLeft CORE's NG SAST, whose results are found in Table 13 in Appendix A, and Figure 13 in Appendix B. The expectation for a commercial tool you need to sign up to use is that performance will generally be better than non-commercial alternatives and to put it bluntly, ShiftLeft CORE's NG SAST fills that expectation perfectly, as with a 100.00% overall TPR, every vulnerability was accurately detected. While there were some false positives, indicated by an overall FPR of 25.27%, this still led to an overall BSA score of 74.43%, the highest overall BSA score by a wide margin. Additionally, only two individual categories had a BSA score of under 50.00%, with SQLi (46.67%) and Weak Encryption Algorithms (44.19%). This is further exemplified by looking at Figure 13, where the top-right corner is pretty much barren and all of the categories are in the middle-to-left side of the top row. The only downside is an unknown run time according to Table 1, but when the results are this good compared to the alternatives, run time is an afterthought if security is the main priority.

6 WORK DISTRIBUTION

In terms of how work was divided up, our group initially was a group of three people. When we were still planning on testing PHP static analysis tools exclusively we came up with the idea that each person in the group would pick one static analysis tool to analyze two GitHub repositories to compare the scan results and then evaluate the effectiveness of each tool. However, a problem arose when one of our group members withdrew from the class, leaving the group with just two people.

As a result, we chose to change the scope of our project, instead using OWASP Benchmark to evaluate static analysis tools that work across a variety of languages, not just PHP. As a result, some of the tools we were previously attempting to use from our initial project milestone became obsolete due to lack of support for those tools in the OWASP Benchmark such as RIPS and OWASP ASST which only work on PHP code, in contrast to the OWASP Benchmark which is described on its website as a Java test suite [1]. However, since VisualCodeGrepper, a tool we initially wanted to use, supports multiple languages, including Java and PHP, we were able to use it with the OWASP Benchmark.

From there, all of the work for the project was done jointly with Kyle predominately focusing on the research aspect, learning about the OWASP Benchmark and first suggesting it to the team, and Miguel mostly focusing on getting the Benchmark to run, as well as finding ShiftLeft CORE's NG SAST and running the Benchmark on that tool.

7 CONCLUSION

When looking at the data collected, we can clearly determine that some static analysis tools work better in regards to specific vulnerability categories than others, and that not all static analysis tools are created equal. While the OWASP Benchmark may not be perfect, as no benchmark truly is perfect, OWASP's goal as stated on the benchmark website is to make application security visible[1]. Our purpose for evaluating this set of static analysis tools is for similar reasons, to allow consumers and people who use static analysis tools to make an informed choice as to which tools should be used, either in general, or for specific use cases that may be more or less important to other users.

REFERENCES

[1] OWASP, "Owasp benchmark." https://owasp.org/www-project-benchmark/.
[2] NIST, "Samate." https://www.nist.gov/itl/ssd/software-quality-group/samate.
[3] P. Nunes, I. Medeiros, J. C. Fonseca, N. Neves, M. Correia, and M. Vieira, "Benchmarking static analysis tools for web security," *IEEE Transactions on Reliability*, vol. 67, no. 3, pp. 1159–1175, 2018.

A TABLES

Table 1. Static Analysis Tool Runtimes

Static Analysis Tool	Runtime
FBwFindSecBugs v1.4.0	0:01:50
FBwFindSecBugs v1.4.3	0:01:58
FBwFindSecBugs v1.4.4	0:04:13
FBwFindSecBugs v1.4.5	0:02:09
FBwFindSecBugs v1.4.6	0:02:02
FindBugs v3.0.1	0:01:32
OWASP ZAP vD-2015-08-24	5:00:00
OWASP ZAP vD-2016-09-05	Time not specified
PMD v5.2.3	0:00:11
SonarQube Java Plugin v3.14	0:05:30
VisualCodeGrepper v2.2.0	Time not specified
ShiftLeft CORE's NG SAST	Unknown

Table 2. FBwFindSecBugs v1.4.0 Results

Category	TPR	FPR	Score
Command Injection	73.81%	76.00%	-2.19%
Insecure Cookie	0.00%	0.00%	0.00%
LDAP Injection	14.81%	28.12%	-13.31%
Path Traversal	84.21%	79.26%	4.95%
SQLi	53.68%	55.60%	-1.93%
Trust Boundary Violation	0.00%	0.00%	0.00%
Weak Encryption Algorithm	74.62%	56.90%	17.72%
Weak Hashing Algorithm	21.71%	0.00%	21.71%
Weak Random Number	100.00%	0.00%	100.00%
XPath Injection	100.00%	100.00%	0.00%
XSS	1.22%	0.00%	1.22%
Overall Results	47.64%	35.99%	11.65%

Table 3. FBwFindSecBugs v1.4.3 Results

Category	TPR	FPR	Score
Command Injection	100.00%	90.40%	9.60%
Insecure Cookie	100.00%	0.00%	100.00%
LDAP Injection	100.00%	90.62%	9.38%
Path Traversal	84.21%	79.26%	4.95%
SQLi	100.00%	91.38%	8.62%
Trust Boundary Violation	0.00%	0.00%	0.00%
Weak Encryption Algorithm	100.00%	45.69%	54.31%
Weak Hashing Algorithm	68.99%	0.00%	68.99%
Weak Random Number	100.00%	0.00%	100.00%
XPath Injection	100.00%	100.00%	0.00%
XSS	0.41%	0.00%	0.41%
Overall Results	77.60%	45.21%	32.39%

Table 4. FBwFindSecBugs v1.4.4 Results

Category	TPR	FPR	Score
Command Injection	100.00%	88.80%	11.20%
Insecure Cookie	100.00%	0.00%	100.00%
LDAP Injection	100.00%	84.38%	15.62%
Path Traversal	96.24%	86.67%	9.57%
SQLi	100.00%	90.52%	9.48%
Trust Boundary Violation	0.00%	0.00%	0.00%
Weak Encryption Algorithm	100.00%	45.69%	54.31%
Weak Hashing Algorithm	68.99%	0.00%	68.99%
Weak Random Number	100.00%	0.00%	100.00%
XPath Injection	100.00%	95.00%	5.00%
XSS	1.22%	0.00%	1.22%
Overall Results	78.77%	44.64%	34.13%

Table 5. FBwFindSecBugs v1.4.5 Results

Category	TPR	FPR	Score
Command Injection	100.00%	88.80%	11.20%
Insecure Cookie	100.00%	0.00%	100.00%
LDAP Injection	100.00%	84.38%	15.62%
Path Traversal	96.24%	86.67%	9.57%
SQLi	100.00%	90.52%	9.48%
Trust Boundary Violation	81.93%	81.40%	0.53%
Weak Encryption Algorithm	100.00%	45.69%	54.31%
Weak Hashing Algorithm	68.99%	0.00%	68.99%
Weak Random Number	100.00%	0.00%	100.00%
XPath Injection	100.00%	95.00%	5.00%
XSS	100.00%	62.68%	37.32%
Overall Results	95.20%	57.74%	37.46%

Table 6. FBwFindSecBugs v1.4.6 Results

Category	TPR	FPR	Score
Command Injection	100.00%	88.80%	11.20%
Insecure Cookie	100.00%	0.00%	100.00%
LDAP Injection	100.00%	84.38%	15.62%
Path Traversal	96.24%	86.67%	9.57%
SQLi	100.00%	90.52%	9.48%
Trust Boundary Violation	100.00%	81.40%	18.60%
Weak Encryption Algorithm	100.00%	45.69%	54.31%
Weak Hashing Algorithm	68.99%	0.00%	68.99%
Weak Random Number	100.00%	0.00%	100.00%
XPath Injection	100.00%	95.00%	5.00%
XSS	100.00%	62.68%	37.32%
Overall Results	96.84%	57.74%	39.10%

Table 7. FindBugs v3.0.1 Results

Category	TPR	FPR	Score
Command Injection	0.00%	0.00%	0.00%
Insecure Cookie	0.00%	0.00%	0.00%
LDAP Injection	0.00%	0.00%	0.00%
Path Traversal	2.26%	1.48%	0.77%
SQLi	53.68%	55.60%	-1.93%
Trust Boundary Violation	0.00%	0.00%	0.00%
Weak Encryption Algorithm	0.00%	0.00%	0.00%
Weak Hashing Algorithm	0.00%	0.00%	0.00%
Weak Random Number	0.00%	0.00%	0.00%
XPath Injection	0.00%	0.00%	0.00%
XSS	0.41%	0.00%	0.41%
Overall Results	5.12%	5.19%	-0.07%

Table 8. OWASP ZAP vD-2015-08-24 Results

Category	TPR	FPR	Score
Command Injection	34.92%	0.00%	34.92%
Insecure Cookie	100.00%	0.00%	100.00%
LDAP Injection	0.00%	0.00%	0.00%
Path Traversal	0.00%	0.00%	0.00%
SQLi	34.56%	0.43%	34.13%
Trust Boundary Violation	0.00%	0.00%	0.00%
Weak Encryption Algorithm	0.00%	0.00%	0.00%
Weak Hashing Algorithm	0.00%	0.00%	0.00%
Weak Random Number	0.00%	0.00%	0.00%
XPath Injection	0.00%	0.00%	0.00%
XSS	28.86%	0.00%	28.86%
Overall Results	18.03%	0.04%	17.99%

Table 9. OWASP ZAP vD-2016-09-05 Results

Category	TPR	FPR	Score
Command Injection	32.54%	0.00%	32.54%
Insecure Cookie	100.00%	0.00%	100.00%
LDAP Injection	0.00%	0.00%	0.00%
Path Traversal	0.00%	0.00%	0.00%
SQLi	58.09%	1.29%	56.80%
Trust Boundary Violation	0.00%	0.00%	0.00%
Weak Encryption Algorithm	0.00%	0.00%	0.00%
Weak Hashing Algorithm	0.00%	0.00%	0.00%
Weak Random Number	0.00%	0.00%	0.00%
XPath Injection	0.00%	0.00%	0.00%
XSS	28.86%	0.00%	28.86%
Overall Results	19.95%	0.12%	19.84%

Table 10. PMD v5.2.3 Results

Category	TPR	FPR	Score
Command Injection	0.00%	0.00%	0.00%
Insecure Cookie	0.00%	0.00%	0.00%
LDAP Injection	0.00%	0.00%	0.00%
Path Traversal	0.00%	0.00%	0.00%
SQLi	0.00%	0.00%	0.00%
Trust Boundary Violation	0.00%	0.00%	0.00%
Weak Encryption Algorithm	0.00%	0.00%	0.00%
Weak Hashing Algorithm	0.00%	0.00%	0.00%
Weak Random Number	0.00%	0.00%	0.00%
XPath Injection	0.00%	0.00%	0.00%
XSS	0.00%	0.00%	0.00%
Overall Results	0.00%	0.00%	0.00%

Table 11. SonarQube Java Plugin v3.14 Results

Category	TPR	FPR	Score
Command Injection	84.92%	87.20%	-2.28%
Insecure Cookie	100.00%	0.00%	100.00%
LDAP Injection	100.00%	100.00%	0.00%
Path Traversal	0.00%	0.00%	0.00%
SQLi	0.00%	0.00%	0.00%
Trust Boundary Violation	0.00%	0.00%	0.00%
Weak Encryption Algorithm	100.00%	0.00%	100.00%
Weak Hashing Algorithm	68.99%	0.00%	68.99%
Weak Random Number	100.00%	0.00%	100.00%
XPath Injection	0.00%	0.00%	0.00%
XSS	0.00%	0.00%	0.00%
Overall Results	50.36%	17.02%	33.34%

Table 12. VisualCodeGrepper v2.2.0 Results

Category	TPR	FPR	Score
Command Injection	44.44%	45.60%	-1.16%
Insecure Cookie	100.00%	100.00%	0.00%
LDAP Injection	0.00%	0.00%	0.00%
Path Traversal	96.24%	88.15%	8.09%
SQLi	52.21%	54.74%	-2.54%
Trust Boundary Violation	32.53%	18.60%	13.93%
Weak Encryption Algorithm	74.62%	0.00%	74.62%
Weak Hashing Algorithm	0.00%	0.00%	0.00%
Weak Random Number	88.53%	18.91%	69.62%
XPath Injection	100.00%	100.00%	0.00%
XSS	0.00%	0.00%	0.00%
Overall Results	53.51%	38.73%	14.78%

Table 13. ShiftLeft CORE's NG SAST Results

Category	TPR	FPR	Score
Command Injection	100.00%	36.00%	64.00%
Insecure Cookie	100.00%	22.97%	77.03%
LDAP Injection	100.00%	0.00%	100.00%
Path Traversal	100.00%	40.62%	59.38%
SQLi	100.00%	53.33%	46.67%
Trust Boundary Violation	100.00%	37.50%	62.50%
Weak Encryption Algorithm	100.00%	55.81%	44.19%
Weak Hashing Algorithm	100.00%	0.00%	100.00%
Weak Random Number	100.00%	0.00%	100.00%
XPath Injection	100.00%	0.00%	100.00%
XSS	100.00%	35.00%	65.00%
Overall Results	100.0%	25.57%	74.43%

B FIGURES

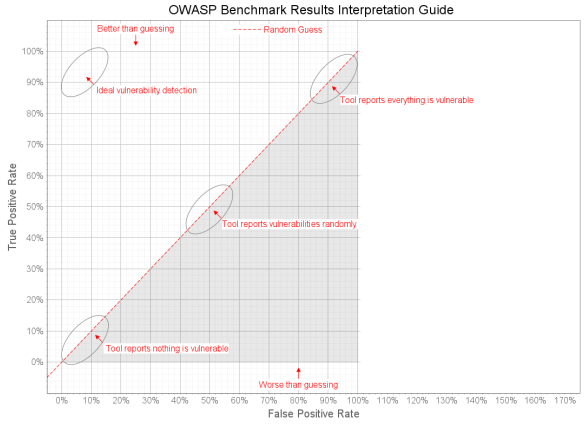


Fig. 1. Provided guide that shows what scorecard graph results mean

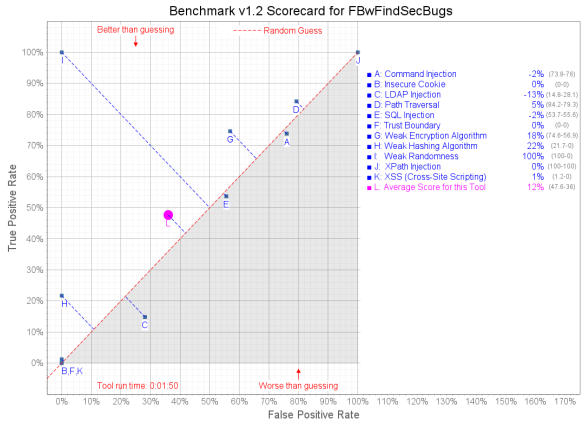


Fig. 2. OWASP Benchmark Scorecard for FBwFindSecBugs v1.4.0

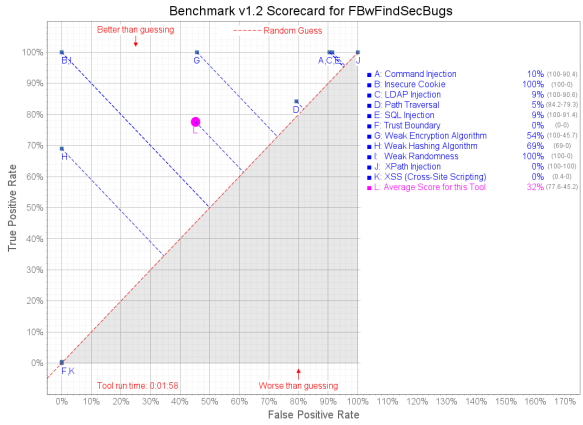


Fig. 3. OWASP Benchmark Scorecard for FBwFindSecBugs v1.4.3

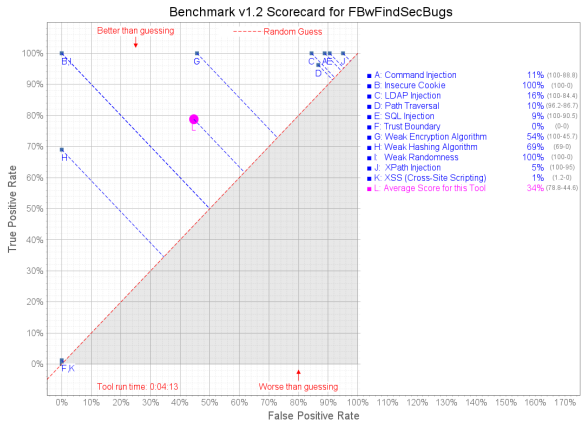


Fig. 4. OWASP Benchmark Scorecard for FBwFindSecBugs v1.4.4

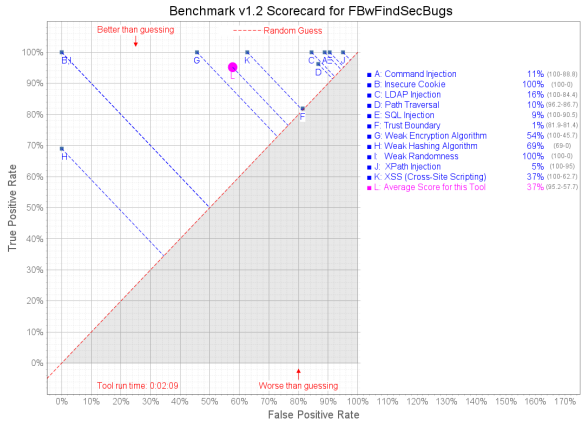


Fig. 5. OWASP Benchmark Scorecard for FBwFindSecBugs v1.4.5

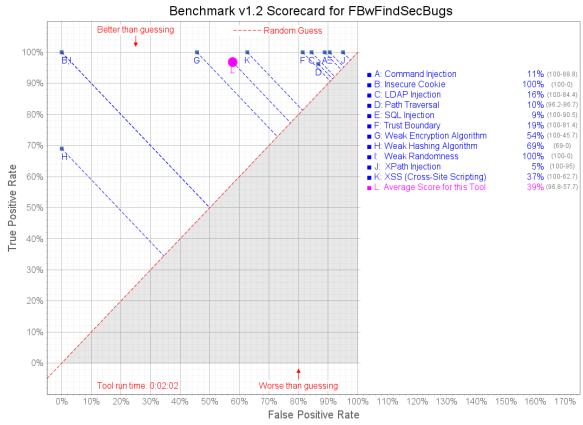


Fig. 6. OWASP Benchmark Scorecard for FBwFindSecBugs v1.4.6

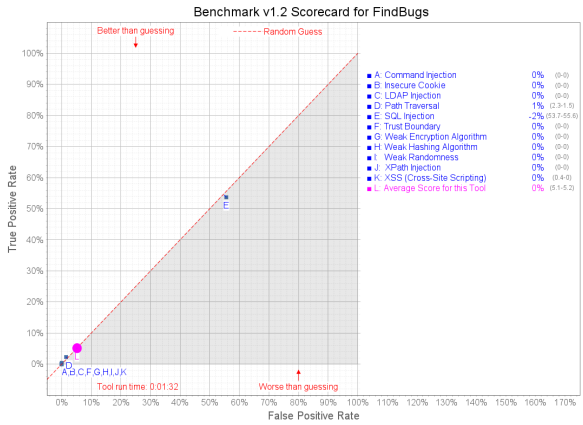


Fig. 7. OWASP Benchmark Scorecard for FindBugs v3.0.1

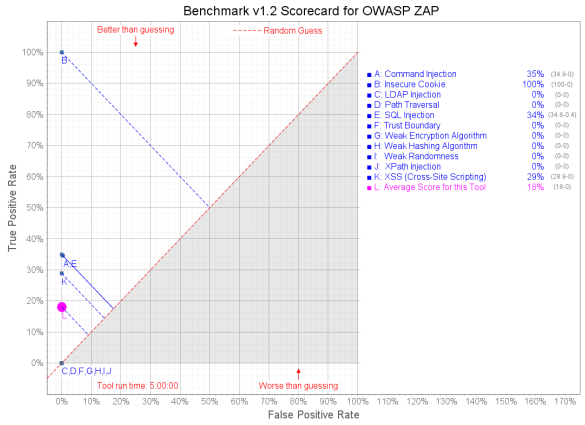


Fig. 8. OWASP Benchmark Scorecard for OWASP ZAP vD-2015-08-24

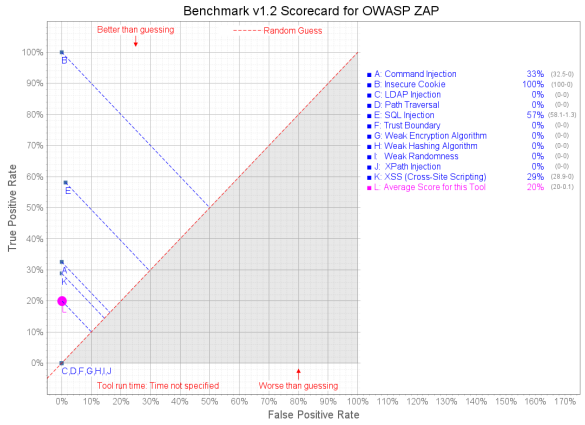


Fig. 9. OWASP Benchmark Scorecard for OWASP ZAP vD-2016-09-05

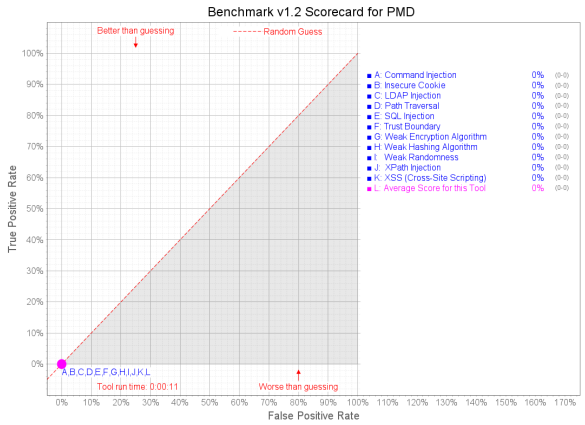


Fig. 10. OWASP Benchmark Scorecard for PMD v5.2.3

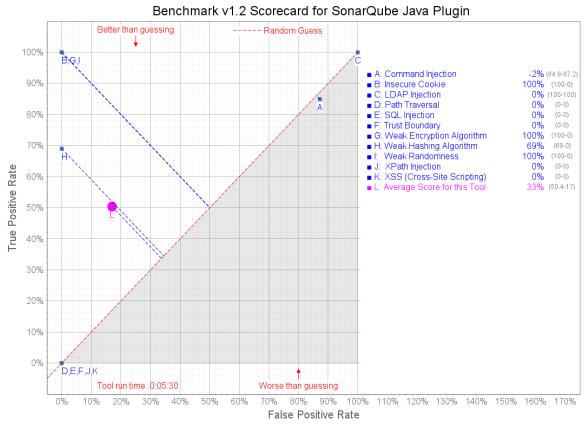


Fig. 11. OWASP Benchmark Scorecard for SonarQube Java Plugin v3.14

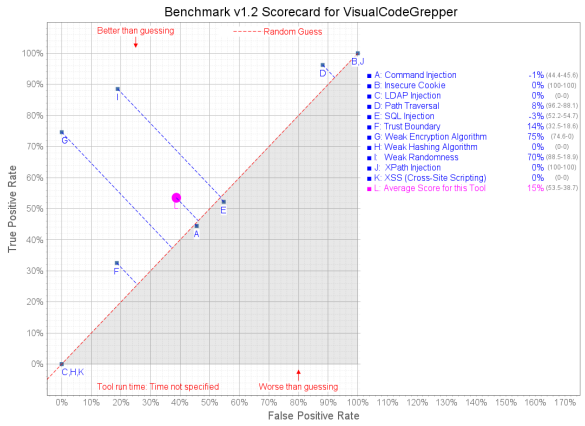


Fig. 12. OWASP Benchmark Scorecard for VisualCodeGrepper v2.2.0

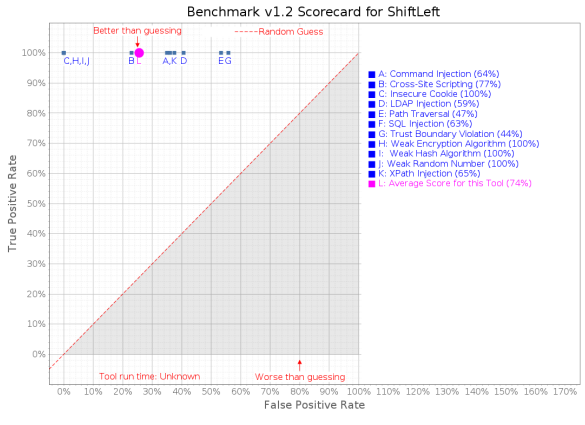


Fig. 13. OWASP Benchmark Scorecard for ShiftLeft CORE's NG SAST