Security Tool Demo Report

Kaiyu Dong, Long Chen, Shaolun Liu

Simon Fraser University
{kaiyu_dong,long_chen_3,shaolun_liu}@sfu.ca

Keywords: Arachni · Vulnerability Scan · DAST

1 Introduction

Arachni is an open-source scanner that focuses on the recon phase of our penetration testing in a different manner than any other tool out there. The creator of the tool, Tasos Laskos developed the tool to address a couple of opposed goals. First, scans can take an excessive amount of time (many hours to even weeks) and as such makes these scans less-than-helpful. That time is lost and makes testing a more drawn-out process. Second, more data and coverage is a good thing, as it enhances accuracy, but it also adds additional time to the test process to complete the necessary transactions.
Laksos developed Arachni to both reduce the amount of time for a scan while allowing the tool to scale such that it is able to process more test vectors efficiently. The timing was improved by employing asynchronous HTTP requests to do the tool's bidding, and by allowing Arachni to scale across a cluster of systems so that they are all processing in parallel. The accuracy goal was enhanced by open-sourcing the tool and using a Ruby framework that can allow anyone to add new or better-optimized tests to the tool. Together, these enhancements make Arachni a formidable scan tool that we should all dive deeper into for improved testing efficacy and timing.
Arachni differs from other scanners in that it makes heavy use of asynchronous HTTP to conduct its scans. This allows each Instance to send HTTP requests to targets in parallel. Because you can have multiple scans in progress at the same time, it greatly improves the speed with which we can go from intitiation to completion. Most other scanners spend considerable time waiting on scan threads to finish. This is made more aggregious given that these scanners often run a set array of scans, and do not tailor their scan list on the fly like Arachni.

2 Using Arachni

The Arachni system is implemented as a series of batch files. Initializing a total scan of a Web asset launches all of the batch files in sequence, each executing a different test vector. This is a series of pen testing attempts covering a full range of exploits, including HTTP GET attacks, SQL injection, code injection, cross-site scripting, and so on. There are a total of 40 modules that provide a

set of passive checks and active attempts. Each module will run through many permutations of possible inputs to test its particular topic of inquiry. All of this work can take a long time to run.
Running the system from the command line results in a text-based report displayed on the screen.

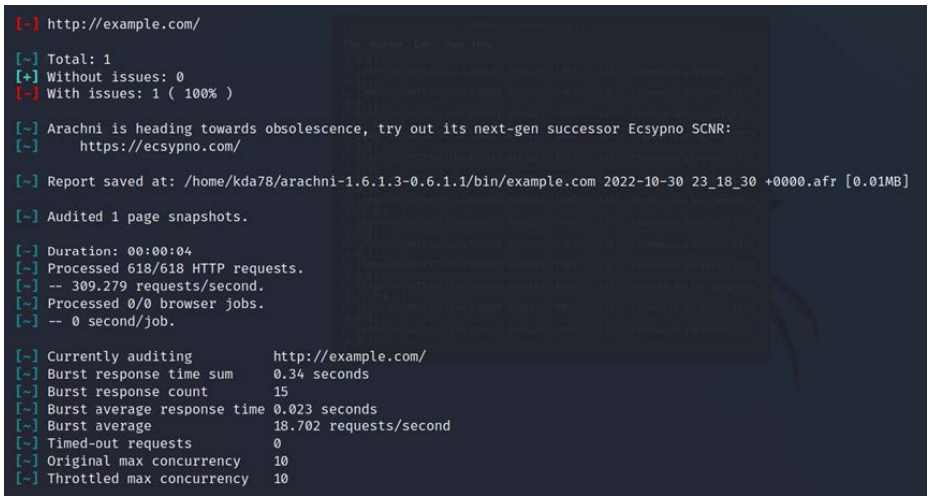

Fig. 1. Running Arachni from the command line

This report is also saved to an AFR file. This is a native binary format – AFR stands for Arachni Framework Report. The file can be converted into various formats, such as XML, plain text, and HTML.
The other method of access to the scanner is through its Web-based GUI. To get the GUI running, you need to open a Command Prompt window, go to the home directory of Arachni on your computer, and enter bin
arachni_web. You will see a message telling you that the listener is running on localhost:<port> - you will see a number instead of <port>. Go to your browser and enter that address in the address bar (e.g., localhost:9292). This opens the Web interface. You need to log in, giving the username admin@admin.admin and password administrator.
This leads to the launch page for the scanner, which gives you a range of options. I left all of the defaults for a test and specified a Direct mode scan of the Arachni website at https://demo.testfire.net/.
As the test run progresses, a stream of feedback messages will pass by in the Command Prompt window.
The Web interface gives a live progress report.
The lower part of the scan report shows the number of problems encountered by type. Clicking on a category gets an explanation of the kind.

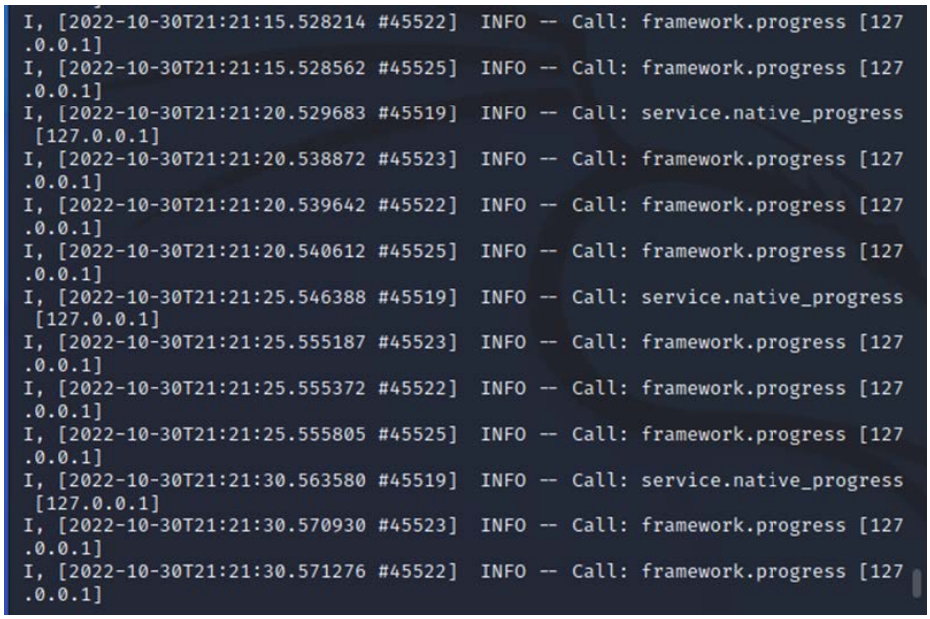

Fig. 2. Feedback messages in the Command Prompt window

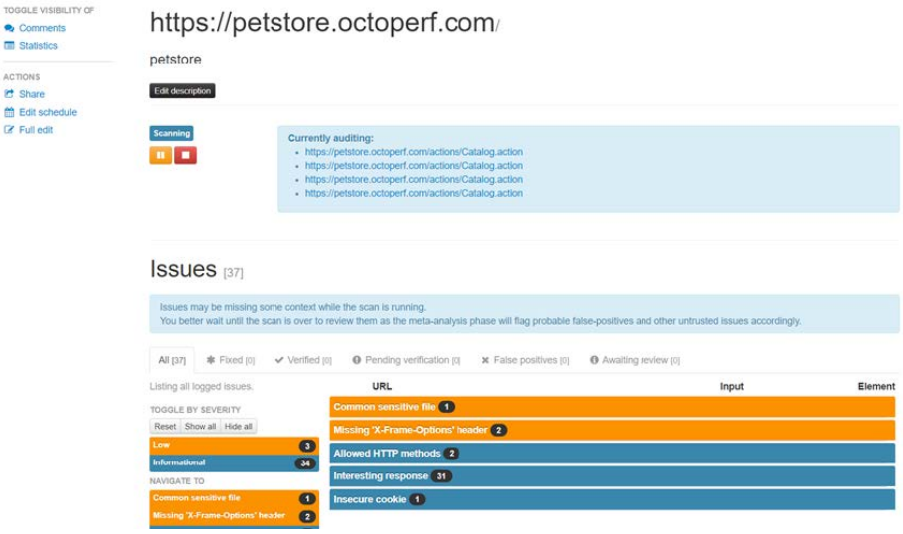

Fig. 3. The live progress report in the Web interface

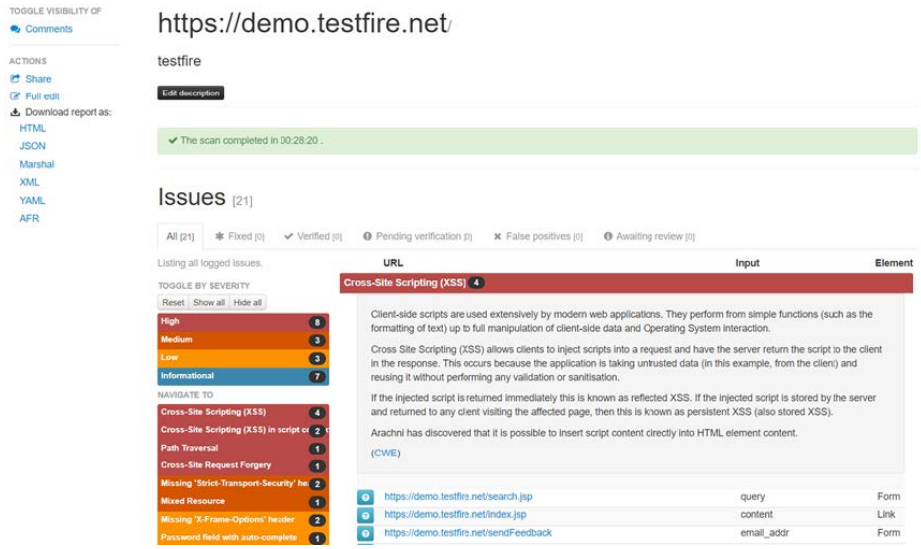

Fig. 4. The number of problems encountered by type

Although creating the scan results in the AFR format is an obscure choice, the utility arachni_reporter can convert that report into HTML format. The results are impressive. Here are the results produced by a scan of the arachni-scanner.com site.
The front tab of the report shows interactive charts based on the scanner's findings.
An Issues tab shows each of the vulnerabilities that were identified.
Further details on each problem can be found by clicking on the line in the Issues report. The system also explains the error and a description of the exploit as given by OWASP.

3 Summary

Arachni is a series of penetration testing exercises implemented as batch files and strung into a single chain that can be launched with one command. However, a vulnerability scanner is precisely that – an automated pen-testing session.
Effective penetration testing is contingent on many things. As current events have revealed, the sheer number of vectors available for attackers to leverage demands comprehensive test suites that automate scans and help us quickly ascertain the risk exposure of a target. Almost as important is the ability to process the copious amounts of raw data and turn it around as actionable intelligence.
Arachni is a fantastic tool to provide these scans, and as an added bonus can form the basis of our detailed reports or deliverables. Because Arachni is an open source and extensible product, it is well supported by the community and offers

Fig. 5. Arachni's report formats are production-ready and a great start on building your deliverables to the customer

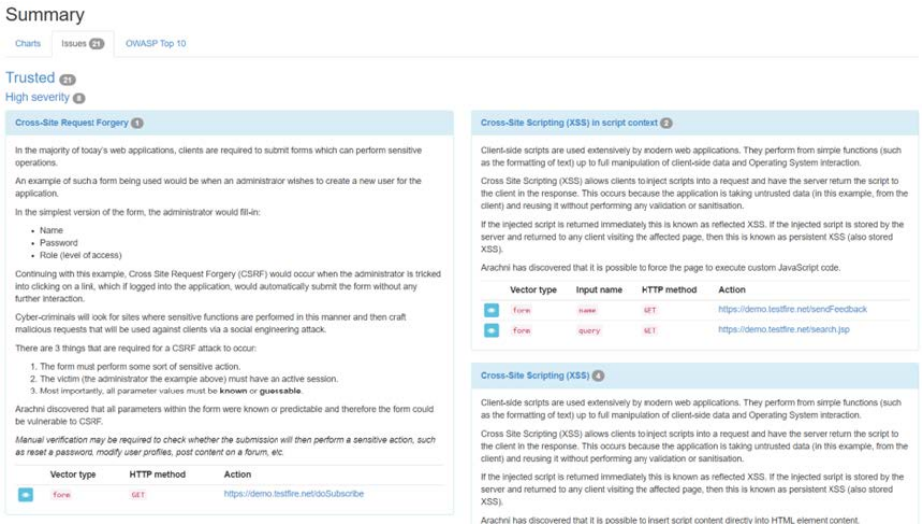

Fig. 6. Detailed Vulnerability information can help quickly educate the customer or set up the next phases

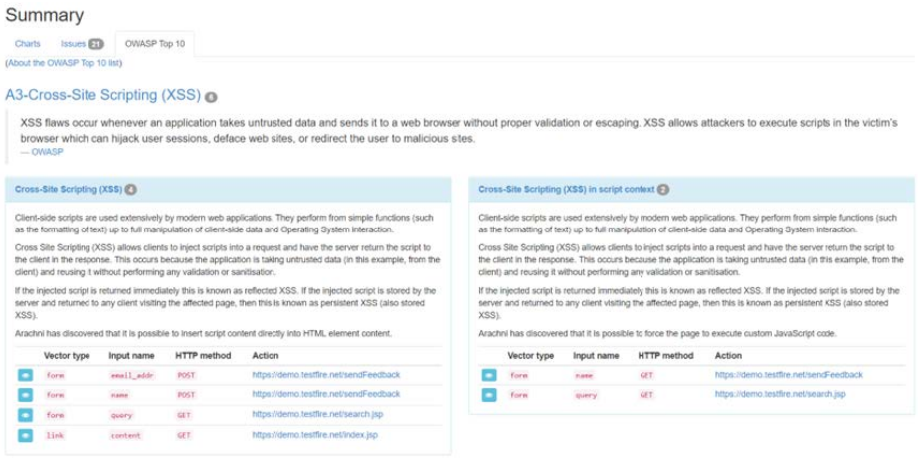

Fig. 7. The description of the exploit given by OWASP

an Ruby framework on which anyone can graft their own extensions or plugins. This system was developed as a student project by one person and promised to be a commercially viable tool. The developer put in a lot of work on this package, and, unlike many enthusiast-created systems, it works and doesn't have any bugs. Many heritage pen-testing tools and vulnerability scanning tools of a similar age to Arachni no longer work or have been made obsolete by advancements in technology. Arachni does not fit into that category – it still works, and its results are insightful.