Interactive Application Security Testing (IAST) – 75% mehr Schwachstellen in Webapplikationen finden

Jörg Sievers
PONTON GmbH
E-Mail: sievers@ponton.de

Abstract: *Herkömmliche Sicherheitstests für Webapplikationen untersuchen den Sourcecode (Static Application Security Testing), ohne den Kontext wirklich zu kennen oder man sucht nach bekannten Sicherheitslücken während der Ausführung der Webapplikation (Dynamic Application Security Testing), hat jedoch dann nicht den Zugriff auf den Source Code. IAST (Interactive Application Security Testing) verbindet beide Verfahren und erzielt erstaunliche Ergebnisse: Keine False-Positives, 100% Erfüllung des OWASP Benchmarks* [1]*, anstatt <30%.*

Problembeschreibung: In den letzten Jahrzehnten hat sich am Sicherheitstesthimmel wenig bewegt. Man setzt auf Tools, die den Source Code durchleuchten, wie SonarQube, PMD, FindSecBugs aus der Open-Source-Welt oder Veracode SAST, Fortify, AppScan Source, um nur einige aus der kommerziellen Welt zu nennen. Ist der Source Code in eine ausführbare Version überführt worden, folgt man dem Mantra der Vermessung der Applikation, um sie dann mit bekannten Sicherheitslücken zu beschießen. Dazu bedient man sich Tools, wie ZAP, Burp, WebInspect. Die Aufwände, die dabei entstehen, sind nicht unerheblich, denn diese Tools wollen beherrscht und justiert werden, sind aber nichts im Vergleich zur Auswertung der Ergebnisse, denn die *False-Positives* müssen aus den Ergebnissen durch Spezialwissen des Sicherheitstesters entfernt werden. Dies ist auch eines der größten Einstiegshürden, warum Sicherheitstests oftmals mit der Kneifzange angefasst werden – man könnte sich irren und daher lässt man es lieber.

Neben dem Spezialwissen, finden automatisierte Tests nur bekannte Fehler und keine neuen [2], egal ob es fachliche oder Sicherheits-Tests sind.

Eine Schwierigkeit kommt noch hinzu: Diese Tests, wenn sie denn kritische Schwachstellen finden, verlangen das der Code neu ausgerollt wird und das schnell. Bei agilen, mehrfach täglich ausgerollten Webapplikationen sicherlich eine überwindbare Hürde, aber für „schwergewichtige" Webapplikationen, die Koordination mit anderen Teams oder gar externen Zulieferern erfordern, kann das Zeitfenster auch mal zu groß und der Einbruch in die Webapplikation schon vollzogen sein.

Bei den jüngst bekannten Sicherheitslücken in Apache Struts, die zu zwei Equifax-Einbrüchen [3] im März und September 2017 geführt haben CVE-2017-9805 (veröffentlicht am 9. September 2017) und CVE-2017-5638 (veröffentlicht am 7. März 2017) war nicht nur diese eine Firma betroffen, sondern viele Systeme, die das Framework in der Version 2 einsetzen. Oracle hat bspw. erst am 22. September 2017 eine Liste zur Verfügung gestellt, welche ihrer Softwareprodukte betroffen sind [4]. Da die Lücken, die zu diesen Einbrüchen geführt haben aber schon im März 2017 bekannt wurden, hatten die Einbrecher also über ein halbes Jahr Zeit Systeme zu finden, die diese Lücke aufwiesen.

Entstehung IAST: Die Technologie beruht auf dem in Java 1.5 (30. September 2004) eingeflossenem JSR-175 (Metadata Facility). IAST „traced" die Web-Applikation zur Laufzeit und liest Informationen zu Klassen, Interfaces, Methoden und Feldern aus [5]. Ausdrücklich sollte Entwicklungstools die Möglichkeit gegeben werden auf diese Informationen zuzugreifen. IAST-Tools klinken sich hier ein und mittels Algorithmen zur Sicherheitsschwachstellenanalyse wird der Code wie bei SAST-Tools durchleuchtet, aber eben zur Laufzeit („Full IAST"). Einige Hersteller nutzen auch den DAST-Ansatz, um zur Laufzeit bekannte CVE-Exploits auszuführen („IAST Light") und besitzen weniger Intelligenz als solche, die den auszuführenden Code wirklich analysieren [6]. Auf Betriebssystemebene könnte man das mit DTrace auf Solaris™ [7] und anderen UNIX®-Derivaten vergleichen, wobei dort die Kernel-Aufrufe analysiert werden, ebenfalls ohne den Code vorher zu instrumentalisieren.

Sicherheitstests als operative Aufgabe: Was zunächst unscheinbar klingt, kann den Umgang mit Sicherheitstests grundlegend ändern, denn:

1. Durch einfaches Einbinden einer Java-Klasse eines IAST-Tools in den Aufruf der eigentlichen Web-Applikation kann jeder Entwicklungsrechner zum aktiven Sicherheitsscanner werden.
2. Entwickler können während der Entwicklung auf gefundene Sicherheitsverletzungen sofort reagieren und müssen nicht von False-Positives herausfiltern, die SAST-Tools meistens liefern. Ihnen wird mittels Metadaten aufgezeigt wo und was als Sicherheitsverletzung erkannt wurde (Stack Trace).
3. DevOps bekommen ein Monitoring-Tool für Sicherheit auf Applikationsebene in die Hand und können auf Abnahme- oder Live-Systemen sofort Maßnahmen ergreifen, um die Sicherheitsverletzungen zu schließen.

Letzteres geht bei einem „Full IAST"-Hersteller so weit, dass man zur Laufzeit den Aufruf schadhafter Stellen zur Laufzeit mittels Unterdrückungsregel verhindern kann!

Im Falle von Equifax und dem Struts 2-Fehler wäre der Angriff erkannt worden [8] und man hätte die Ausführung der schadhaften Stelle(n) unterbinden können.

Proof-of-Concept beweist Vorteile: In einem von mir durchgeführten Proof-of-Concept (PoC) einer Webapplikation auf Basis von Apache Wicket, Hybernate und Spring wurden mittels herkömmlicher DAST-Tools in einem Sicherheitsaudit einer externen Firma zwei Sicherheitsverletzungen gefunden. Ein „IAST-Light"-Tool fand auf derselben Applikation 10 Verletzungen und ein „Full-IAST"-Tool 21. Da „Full-IAST" bis hin zum Backend der Web-Applikation die Aufrufe prüfte, wurden hier auch sehr tief liegende Verletzungen (Passwort-Speicherungs-Algorithmus) gefunden. Zudem wurde von Entwicklern das einfache Beheben von angemerkten Schwachstellen gelobt. Der PoC bestätigte ebenso die 100% Abwesenheit von False-Positives, denn alle ermittelten Schwachstellen waren wirkliche Programmierlücken. Die untersuchten IAST-Tools boten alle eine gute Weboberfläche für das Monitoring der gefundenen Schwachstellen und brachten zudem noch die Analyse der verwendeten Bibliotheken mit und verglichen diese gegen bekannte CVE-Verletzungen und deren Versionsnummern mit denen aktuell verfügbaren.

Performanz: Alle Hersteller betonen, dass man deren IAST-Lösung auch in Live-Umgebungen einsetzen kann, sofern es sich nicht um sehr zeitkritische Anwendungen, wie etwa High Frequency Trading, handelt. In dem PoC wurde das ebenfalls so wahrgenommen, da es sich um eine normale Geschäftsanwendung handelt. Der erste Start, der auch die Analyse der genutzten Bibliotheken beinhaltet war spürbar langsamer, jedoch das bekannte Verhalten der Applikation wurde nicht beeinträchtigt.

IAST füllt für Web-Applikationen die Lücken, die DAST und SAST hinterlassen sehr gut auf und unterstützt zudem den DevOps-Entwicklungsansatz und hebt Sicherheitstests auf eine Ebene, wo Spezialwissen zur Untersuchung einer Web-Applikation nicht mehr benötigt wird, lediglich die wirkliche Behebung der Schwachstellen muss natürlich von Entwicklern durchgeführt werden.

In den „Magic Quadrant for Application Security Testing"-Bericht von Gartner [9] findet man eine Auswahl an Herstellern und deren Bewertung.

Quellenverzeichnis:

[1] „OWASP Benchmark Project," [Online]. Available: https://www.owasp.org/index.php/Benchmark. [Zugriff am 17 9 2017].

[2] D. Graham und M. Fewster, Software Test Automation, Addison-Wesley, 1999.

[3] ds@heise.de, „Equifax soll früheren Hack verheimlicht haben," Heise Medien GmbH & Co. KG, 19 9 2017. [Online]. Available: https://www.heise.de/newsticker/meldung/Equifax-soll-frueheren-Hack-verheimlicht-haben-3835052.html. [Zugriff am 23 9 2017].

[4] „Oracle Security Alert CVE-2017-9805 Products," Oracle Inc., 22 9 2017. [Online]. Available: http://www.oracle.com/technetwork/security-advisory/cve-2017-9805-products-3905487.html. [Zugriff am 23 9 2017].

[5] S. M. I. Joshua Bloch, „JSR 175: A Metadata Facility for the JavaTM Programming Language," 30 09 2004. [Online]. Available: https://jcp.org/en/jsr/detail?id=175. [Zugriff am 12 10 2017].

[6] S. G. Matthias Rohr, „IAST: A New Approach for Agile Security Testing," 26 11 2015. [Online]. Available: https://blog.secodis.com/2015/11/26/the-emerge-of-iast/. [Zugriff am 12 10 2017].

[7] Oracle, Inc., „Observability," [Online]. Available: http://www.oracle.com/technetwork/server-storage/solaris11/technologies/dtrace-1930301.html. [Zugriff am 12 10 2017].

[8] C. S. Jeff Williams, „We are Seeing Ongoing Struts 2 Attacks," 10 4 2017. [Online]. Available: https://www.contrastsecurity.com/security-influencers/what-we-are-seeing-ongoing-struts-2-attacks. [Zugriff am 12 10 2017].

[9] A. T. Dionisio Zumerle, „Magic Quadrant for Application Security," Gartner, 2017.