Aufdecken von Sicherheitsschwachstellen in der Software durch statische Codeanalyse

von Dipl.-Ing. Royd Lüdtke

Moderne Softwareapplikationen interagieren zunehmend via Intranet oder Internet miteinander und bieten oft zudem die Möglichkeit ihre Funktionen über das Netz kontrollieren zu lassen. Damit gewinnt, neben der Forderung nach funktionaler Sicherheit, die Forderung nach Sicherheit vor Angriffen auf eine Applikation zunehmend an Bedeutung. Dem sehen sich sowohl die Softwareentwicklung als auch die Softwarebeschaffung (Software Supply Chain Security) vieler Unternehmen ausgesetzt.

Wie kann die statische Codeanalyse die Softwareentwicklung unterstützen? Sofern alle Softwarekomponenten eines Projektes ausschließlich Eigenentwicklungen sind, können etwaige Sicherheitsschwachstellen mit Hilfe von Werkzeugen zur statischen Quellcodeanalyse mit hoher Erfolgsquote aufgedeckt und bereits frühzeitig im Entwicklungsprozess eliminiert werden. Weitaus problematischer ist die sicherheitstechnische Überprüfung von Softwareprojekten, wenn Komponenten von Drittanbietern in die Applikationen mit eingebunden sind. Abschätzungen zufolge trifft dies für 95% aller proprietären Projekte zu.

Man unterscheidet drei Kategorien von Drittanbieter-Software:

- Open Source Software (OSS) – Diese Software unterliegt einem offenen Entwicklungsprozess und wird zur Integration in den eigenen Quellcode lizensiert.
- Commercially off-the-shelf software (COTS) – Software, die von einem Anbieter bezogen wurde. Meist ist keine spezielle Adaptierung notwendig. Falls Open Source -Komponenten enthalten sind, wird dies gegenüber dem Lizenznehmer eher selten offengelegt. COTS-Software wird vornehmlich als Binärdatei ausgeliefert.
- Speziell beauftragte Software – Entwickelt zur Einbindung in eine bestimmte Applikation oder Applikationsfamilie. Die Verwendung von Open Source-Komponenten wird oft nicht oder nur unvollständig offengelegt. Zumeist werden solche Bibliotheken als Binärdatei ausgeliefert.

Oft ist über die Qualität von lediglich als Binärdatei gelieferten Komponenten wenig bekannt. Vielfach fehlt auch die Information darüber, welche Open Source-Komponenten der Drittanbieter seinerseits in seine Komponente integriert hat. Moderne Werkzeuge zur statischen Binäranalyse können in solchen Fällen helfen, sowohl 0-Day- als auch N-Day-Sicherheitsschwachstellen aufzudecken bzw. auszuweisen.

0-Day-Sicherheitsschwachstellen sind Probleme, die durch Analysen neu aufgedeckt werden und daher noch unveröffentlicht sind. Diese sollten möglichst bald beseitigt werden, stellen allerdings in der Regel noch keine unmittelbare Gefahr dar, da

Angreifer, wegen fehlender Kenntnis über das Vorhandensein des Sicherheitsproblems, dieses wahrscheinlich nicht nutzen können.

N-Day-Sicherheitsschwachstellen sind dagegen bekannte, in öffentlichen Datenbanken ausgewiesene Sicherheitslücken. Von N-Day-Schwachstellen geht ein besonders hohes Gefahrenpotenzial aus. Wurde in eine Applikation eine Open Source Bibliothek eingebunden, für die hochriskante N-Day-Sicherheitsprobleme bekannt sind, ist das Programm zielgerichteten und wahrscheinlich auch erfolgreichen Angriffen ausgesetzt, sofern die Verwendung dieser Bibliothek in der Applikation vermutet werden kann. In diesen Fällen ist unbedingt eine unverzügliche Problembehebung erforderlich.

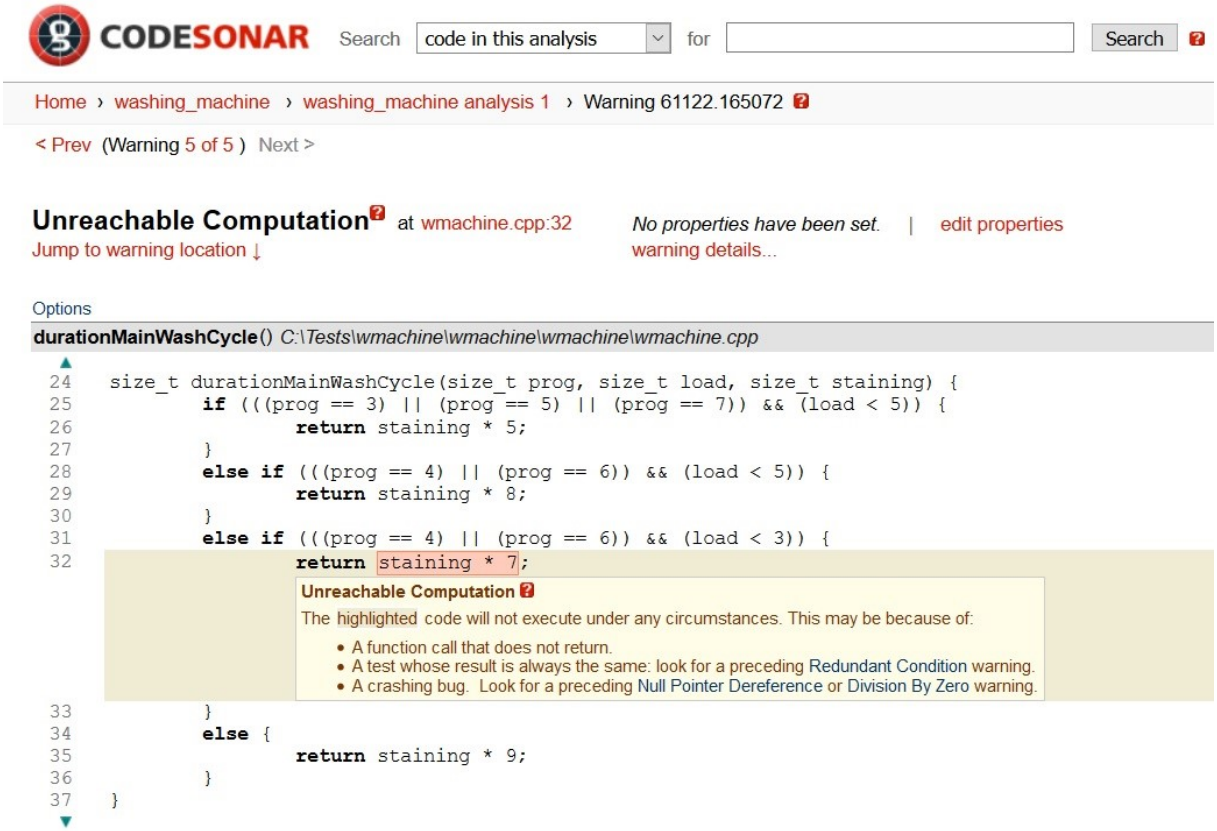

Abbildung 1: Bei selbst erstelltem Code können Schwachstellen durch statische Quellcode-analyse (hier GrammaTech CodeSonar) aufgedeckt werden

Zur Aufdeckung von 0-Day-Sicherheitsschwachstellen stehen Werkzeuge zur statischen Binäranalyse zur Verfügung, die insbesondere auf Sicherheitsanalysen spezialisiert sind (Static Analysis Security Testing (SAST)). Die von diesen Werkzeugen genutzten Verfahren sind hochkomplex und stets abhängig von der „Instruction Set Architecture“ (ISA), welche der Binärdatei zugrunde liegt, der ursprünglichen Programmiersprache, der Datenwortbreite und nicht zuletzt dem Optimierungsgrad des Compilers. Die Fehleraufdeckungsquote solcher Werkzeuge ist hoch, erreicht allerdings generell nicht die der Quellcodeanalyse, da z.B. die Abgrenzung von Variablen, Arrays etc. auf dem Stack nicht unproblematisch ist.

Zur Ermittlung der N-Day-Schwachstellen ist es nötig eine „Software Composition Analysis“ (SCA) durchzuführen. Ist die Zusammensetzung der Binärdatei bekannt, kann in einschlägigen Datenbanken, z.B. der US-amerikanischen National Vulnerability Database (NVD) oder der Open Source Vulnerability Database (OSVDB) nachgeschlagen werden, ob bereits Sicherheitsschwachstellen für die darin enthaltenen Bibliotheken gelistet sind.

Werkzeuge, die eine wirklich tiefgehende Komponentenanalyse von Binärdateien durchführen können, sind erst eine relativ kurze Zeit am Markt verfügbar. Eine Herausforderung ist die zum Teil hohe Verschachtelungstiefe eingebundener Bibliotheken. Diese können von weiteren Bibliotheken abhängig sein, die sich wiederum weiterer Bibliotheken bedienen.

Eine einfache Mustererkennung, beispielsweise, mittels Abgleichs von Hash-Werten, hat sich als alleiniges Kriterium als unzureichend erwiesen, da identischer Quellcode, übersetzt durch unterschiedliche Compiler oder Compiler-Derivate, auch für gleiche Zielplattformen zu stark voneinander abweichenden Binärausgaben führt.

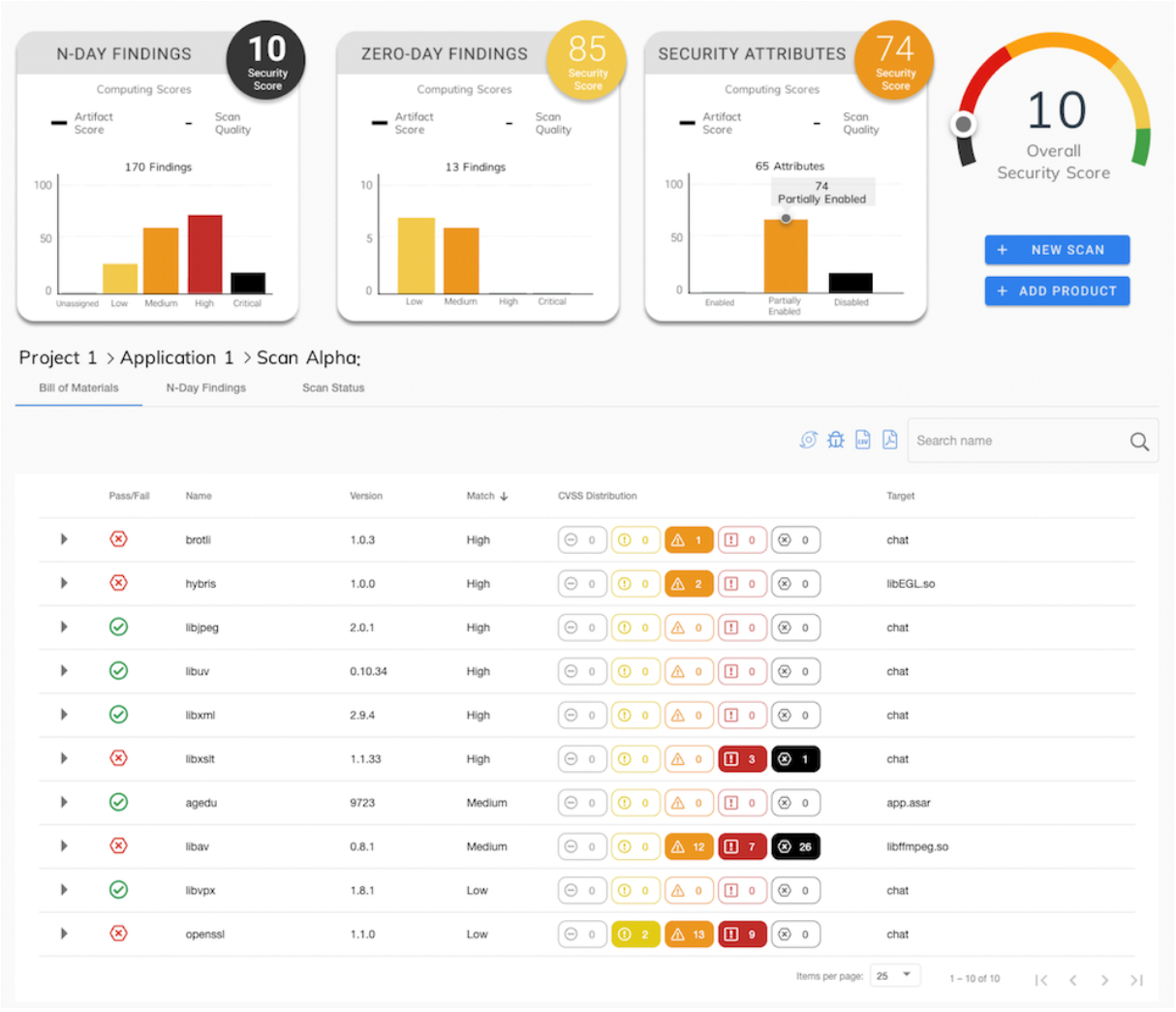

Abbildung 2: Moderne Werkzeuge wie GrammaTech CodeSentry führen eine statische Binärcodeanalyse aus und melden Sicherheitsschwachstellen

Früher war daher lediglich eine wenig zuverlässige, oberflächliche Analyse üblich. Hierbei wird eine Datei auf bekannte, eingebettete Strings hin untersucht. Mit sehr viel Glück findet sich sogar ein String, der die Komponentenbezeichnung inklusive einer Versionsbezeichnung enthält. In Ausnahmefällen, sofern Debug-Informationen enthalten sind, können auch Variablennamen in der Symboltabelle einen Hinweis auf die Identität einer Komponente liefern.

Zur Durchführung einer tiefgehenden Komponentenanalyse muss das Werkzeug durch Bewertung zusätzlicher Indizien die enthaltenen Anteile identifizieren. Neben der bereits erwähnten Erhebung und dem Vergleich von Hash-Werten, sowie der Suche nach bekannten Strings und Namen in der Symboltabelle, werden dann auch die Programmabläufe selbst zur Bewertung herangezogen. Das Werkzeug muss dazu das dem Binärcode zugrunde liegende „Instruction Set“ (ISA) erkennen und befähigt sein, Grundstrukturen des Programms abstrahieren zu können. Lassen sich z.B. Funktionen abgrenzen, können deren Abhängigkeiten voneinander (Call Graph) einen wertvollen Hinweis auf die Komponente selbst liefern. Weitere aussagekräftige Muster können sich durch Kontrollflussanalysen der einzelnen Funktionen ergeben.
Hier wird deutlich, dass, bedingt durch etwaige vielfältige Verschachtelung verschiedener Komponenten, absolute Aussagen über deren Existenz in der Binärdatei kaum mehr möglich sind. Vielmehr müssen intelligente Algorithmen die zusammengetragenen Indizien in Bezug zueinander bewerten, um daraus die Wahrscheinlichkeit für das Vorhandensein einer bestimmten Komponente in der zu Untersuchenden Datei ableiten zu können. Zur Verbesserung der Ergebnisqualität, werden die dafür eingesetzten Algorithmen durch maschinelles Lernen fortwährend optimiert.

Wie können die oben vorgestellten Verfahren in der Softwarebeschaffung eingesetzt werden?
Ist geplant eine Software in einem Unternehmen einzusetzen, spielen je nach Art und vorgesehenem Einsatzgebiet, Datenschutz- und Lizenzrechtaspekte eine wichtige Rolle. Idealerweise sollten daher die statische Sicherheitsanalyse (SAST) sowie die „Software Composition Analysis“ (SCA) ein fester Bestandteil des Beschaffungsprozesses sein.

Einerseits geben die Analysen Auskunft über etwa in der Software enthaltene, gefährliche N-Day- und 0-Day-Sicherheitsschwachstellen. Andererseits ermöglicht die Liste über die in der Software enthaltenen Bibliotheken (Software Bill Of Material (SBOM)) eine lizenzrechtliche Bewertung und vermindert dadurch zusätzliche Risiken.

Der Einsatz statischer Analysetechniken zur Aufdeckung von Sicherheitsproblemen erhöht den Schutz von Softwareapplikationen gegen Angriffe erheblich. Durch die zunehmende Einbindung von Drittanbieter-Software in proprietäre Softwareprojekte, kommen diese Techniken vermehrt auch zur Prüfung von Binärdateien zum Einsatz und werden in naher Zukunft ein unverzichtbarer Bestandteil des Entwicklungsprozesses sein.

Autor:

Diplom-Ingenieur Royd Lüdtke leitet den Bereich Pre-Sales und Support für Statische Codeanalysetools im deutschsprachigen Raum. Royd Lüdtke hat umfangreiche Berufserfahrung als Applikationsingenieur und Berater bei einer Vielzahl von Firmen und Institutionen wie New Era Of Networks, Sybase, Rogue Wave Software und dem Fraunhofer Institut. Lüdtke studierte in Dortmund Elektrotechnik und Energietechnik, hält mehrere Patente und ist Autor von Veröffentlichungen im IT-Bereich.

Links:

Statische Quellcodeanalyse: GrammaTech CodeSonar:
https://www.verifysoft.com/de_grammatech_codesonar.html

Binärcodeanalyse: GrammaTech CodeSentry:
https://www.verifysoft.com/de_grammatech_codesentry.html

Aufdecken von 0-Day- und N-Day-Sicherheitsschwachstellen durch statische Codeanalyse (Video): https://www.youtube-nocookie.com/embed/ckIESnwoUTY