Schlussbericht:

RIOT: Rapid Indexing of (Network) Traffic

Suchindex für Netzwerkverkehre

Alexander Wamack

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS19S35 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Nach wie vor sind Computer-Netzwerke das Ziel von Cyber-Angriffen. Wie kann man sich oder andere davor schützen? In einer idealen Welt wäre das Mittel der Wahl natürlich, alle Sicherheitskonzepte und Richtlinien konsequent umzusetzen. Die Realität aber ist eine andere: große, historisch gewachsene und unter Umständen schlecht dokumentierte Netzwerk-Topologien können nicht ohne Einschnitte des Betriebs auf einen entsprechenden Sicherheitsstandard gebracht werden. Schlimmer noch: Wie kann man sich gegen noch nicht *öffentlich bekannte* Sicherheitslücken – wie beispielsweise CVE-2020-1350 – schützen? Die Antwort ist erschreckend: Erst einmal überhaupt nicht. Man kann jedoch versuchen, durch gezieltes Monitoring des Netzwerk-Perimeters und schnelles Handeln im Ernstfall nach einem erfolgreichen Angriff dessen Auswirkung einschätzen und Gegenmaßnahmen einleiten. Hier fügt sich `NYGMA` mit dem im Rahmen des Prototypefund implementierten Erweiterung `RIOT` in die Liste der Hilfswerkzeuge ein.

Gängige Lösugen aus dem Network Security Monitoring (NSM) und Intrusion Detection (IDS) Bereich erkennen über Muster und Anomalien Angriffe in Netzwerken. Um auf diese angemessen reagieren zu können müssen so viel Informationen wie möglich zur Verfügung stehen. Die Observation des Angriffs bedeutet hier meist eine erfolgreiche Infiltration der Infrastruktur des Opfers die zeitlich u.U. bereits in der Vergangenheit liegt und anderen noch nicht bekannten Mustern gefolgt ist. RIOT stellt eine zuverlässige Datenbasis für forensische Post-Incident Analysen wie Retro-Hunting und Ereignis-Assessment zur Verfügung.

Die am weitesten verbreiteten Open-Source-Software (OSS)-Lösungen `Suricata` für regelbasierte Warnmeldungen und `Zeek` für Metadatenerfassung stellen zur Zeit die Grundlage für ein OSS-basiertes Netzwerk-Sicherheits-Monitoring dar.

Beide Systeme bieten allerdings nicht – oder nur sehr eingeschränkt – die Möglichkeit, auf Rohmaterial zurückzugreifen. Auch eine Funktion für das wiederholte und schnelle Beantworten von Suchanfragen bietet keine der beiden Lösungen. Besonders im Falle von reinen Warnmeldungen fehlen dem*der Analyst*in wichtige Kontextinformationen, um die Relevanz (false-positive?) und Kritikalität (true-positive) eines Ereignisses schnell und korrekt beurteilen zu können. Hier setzt `RIOT` an: `RIOT` ergänzt `Suricata` und `Zeek` um die

Möglichkeit, aus dem bestehenden Datenbestand Informationen im Rohformat gezielt abfragen zu können und weitere Erkenntnisse zu gewinnen oder tiefer gehende Analysen zu ermöglichen.

Das Projekt wurde in der Programmiersparache `C++20` implementiert. Die Projektplanung erfolgte in Absprache mit dem Prototype Fund jeweils wöchentlich eine Woche im Voraus sowie mit einem kurzen Rückblick der vergangenen Woche.

Die Implementierung selbst erfolgte angelehnt an ein Test-Driven-Development: Für jede wichtige Funktionalität wurden einzeln ausführbare Unit-Tests implementiert. Die Tests wurden zusammen mit der tatsächlichen Implementierung entworfen.

Zusätzlich wurden auch Test-Treiber für ein Fuzzing der Komponenten bereitgestellt. Die Treiber bzw. Fuzzer basieren auf `libFuzzer` und `honggfuzz`. Durch die frühe Integration der Fuzzing-Komponenten in Kombination mit den Unit-Tests konnte parallel zur Entwicklung eine QA (Quality Assurance) des `C++`-Codes erfolgen.

Die wichtigsten bearbeiteten Meilensteine waren hierbei:

- Evaluierung der Primär Index Datenstruktur (B+Trees und RedBlack Trees)
- Implementierung / Integration Unit-Test Framework
- Entwurf Dateiformat für Indexstruktur
  - Serialisierung der Indexstruktur
  - Komprimierung der Indexstruktur (Integer Compression)
  - Integration von SIMD Integer Compression Bibliotheken
  - Implementierung Fuzzing-Tests
- Erweiterung des PCAP-Parsers
  - Fehlerbehebung
  - Unit-Tests
  - Erweiterung TCPO/UDP IPV4/IPV6 anhand der Test Daten
  - Implementierung Fuzzing-Tests
  - Erweiterung DNS Parser und Fehlerbehebung
- PCAP slicing und reassembly
  - Unit-Tests
- Desgin Abfragesprache
- Desing der Query (Abfrage) - Engine (API)
  - Implementierung Fuzzing-Tests
- Integration Intel-Hyperscan für Regexp-Index
  - Support für die Abfragesprache
  - Sprache zum Einlesen von Mustern bzw. Indicators of Compromise (IOC)
  - Anwenden von Mustern auf DNS-Namen
- Beispiele und Dokumentation
  - Befehlszeilen Tool zum indizieren nach IP-Addressen und Ports
  - Befehlszeilen Tool zum indizeiren nach Mustern
  - Grafische Benutzeroberfläche für die Demoweek
- Evaluierung Fuzzing und Fehlerbehebung
- Benchmarking der Index-Algorithmen
  - hinsichtlich der Komprimierungsraten

  - hinsichtlich der Geschwindigkeit
- Demo-App und Show-Case
  - Design UI
  - Aufbereitung von Testdaten
  - "Regexp-Index" Workflow
  - Videos für die Demoweek
- Während der Planung ausgeschlosse Milestones
  - Spezialisiertes Format zum Speichern von Netwerkdaten (CCAP)
  - Live Capture

Beitrag des Projektes zu den Zielen der Förderinitative „Software-Sprint"

Die Zielgruppe von `NYGMA/RIOT` sind Netzwerkadministrator*innen und Cyber-Security-Analyst*innen, die bereits mit `Suricata` oder `Zeek` arbeiten und eine Lösung für die Aufbereitung von Computer-Netzwerkdaten benötigen. Konkrete Anwendungsfälle reichen von der Fehlersuche in Netzwerken, leistungsorientierten Optimierungen oder forensischen Untersuchungen bis hin zu Analysen nach oder während eines erfolgreichen Angriffs (Incident-Response). Auch in Schulungen zu Netzwerken allgemein kann `NYGMA/RIOT` eingesetzt werden.

Die stärke von `NYGMA/RIOT` liegt darin, dass es kein einzelnes Programm sondern eine Bibliothek ist die auf den konkreten Einsatzzweck – also das eigene Netzwerk – umfassend angepasst werden kann. Auch wird keinerlei Annahme über die Daten selbst getroffen. Zum Export der Daten werden Standardformate verwendet. Damit läßt sich die Software komfortabel integrieren.

Das Projekt verbindet den Schutz öffentlicher Infrastruktur durch eine Open-Source Lösung, Verbesserung der Tooling-Fähigkeiten bei Cyber-Defense Fragestellungen und den Schutz der Privatspähre durch ein Basisframework für weitere Analysemöglichkeiten mit den Themen der Open-Knowledge-Foundation insb. des Prototypefunds.

Ausführliche Darstellung der Ergebnisse

Im Rahmen der Prototypentwicklung lag der Focus ausschließlich auf der Implementierung der Index-Algorithmen um bereits bestehende Datenstände im Standard-PCAP Format durchsuchbar zu machen. Zusätzliche wurde für die Abschluss-Demonstrations eine grafische Oberfläche implementiert, die den Einsatz aus Sicht einer forensichen Untersuchung ermöglicht.

Die zunächst geplante Entwicklung eines angepassten Speicherformats als auch die dazu notwendige Implementierung der Live-Erfassung von Netzwerkkarten wurde aufgrund des Umfangs des Projekts nach Rücksprache zurückgestellt.

Alle während der Projekzeit besprochenen Meilensteine wurden erreicht und erfolgreich abgeschlossen. Das kommandozeilen basierte Tool `ny` ermöglicht die Erstellung eines Suchindexes für vorhandene PCAP-Daten beliebiger Größe. Die Testdatensätze umfassen Einzeldateien mit einer Größe zwischen 50GB und 100GB je Datei. Das ebenfalls kommandozeilen basierte Tool `t3` erlaubt die Indezierung der Daten mittels selbst definierter Such-Muster unter Verwendung der Open-Source Software `Intel-Hyperscan`. Das Tool `the-0wls` dient zur

grafischen Veranschaulichung der Index-Daten, zur Abfrage des Indexes und zum wiederholten Re-Analysieren des Datenbestandes mit dem Muster-Index durch `t3`.

Das Projekt wurde als Software-Bibliothek entwickelt, die beschriebenen Werkzeuge demonstrieren jeweils nur einen Teilaspekt des eigentlich möglichen Funktionsumfangs. Idealerweise wird die Verwendung auf die jeweilge Netwerkstruktur angepasst. Hierin liegt auch die Stärke der implementierten Software: hohe Anpassbarkeit, keine komplizierten Sofware abhängigkeiten.

Für die Abfrage des Such-Indexes wurde eine kleine domänenspezifische Sprache implementiert die die normalen Menge-Operationen aus der Mathematik implementiert: Schnittmengenbildung, Vereinigungsmenge und Differenz. Die Bibliothek ermöglicht auch ein Abfrage des Invertierten-Indexes als auch eine spezielle Rückwärts-Suche, die es ermöglicht zusammengehörige oder ähnliche Daten auf einmal abzufragen. Diese Funktionalität ist nur in der Bibliothek enthalten und noch nicht über die Sprache selbst benutzbar.

Die Rohdaten wurden hierbei von folgenden öffentlichen Quellen benutzt:

- https://www.netresec.com
- https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

Da `RIOT` für das Bearbeiten von großen Datenbeständen entwickelt wurde lag besonderer Schwerpunkt in der Komprimierung des Such-Indexes. Hierbei wird die zu analysierende Datei in virtuelle Segmente von je maximal 4GB unterteilt. Der Index selbst besteht aus einer Abbildung von Merkmalen wie IP-Addressen, TCP/UDP-Port Nummern oder beliebigen Benutzer definierten Identifikationsnummer zu Offset-Listen in die ursprüngliche Datei. Diese Offset-Listen eigenen sich aufgrund ihres Aufbaus besonders gut für die Komprimierung mittels in der Literatur bezeichneten “Integer-Compression-Algorithms”. Bei der Analyse der Testdaten konnte eine Größe per 4GB Segment Index Daten zwischen 25MB im komprimierten Fall und 60MB im unkomprimierten ermittelt werden.

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Die Vision ist, durch die Kombination von `Suricata`, `Zeek` und `NYGMA/RIOT` die Cyber-Abwehrmöglichkeiten *für alle* zu verbessern. Da Monitoring-Lösungen im Herzen des eigenen Netzwerks platziert werden, ist es eigentlich nahezu unumgänglich, eine Open-Source-Lösung einzusetzen. Diese kann von unabhängigen Spezialist*innen eingesehen werden um sicherzustellen, dass keine Daten “abfließen” und die Software selbst gängigen Sicherheitsvorschriften genügt. Bei kommerziellen Black-Boxen ist ein (mehr oder weniger eingeschränkter) Fernzugriff oder Exfiltration nie auszuschließen, teilweise sogar verpflichtender Gegenstand eines Wartungs- oder Analysevertrags. Datenschutz- und Sicherheitsinteressen widersprechen dem selbstredend.

Beim Verarbeiten der Daten werden spezielle Befehle (SIMD: SSE und AVX) der Intel-X86 Prozessoren verwendet. Eine Portierung der Software für die 64Bit-ARM Architektur (aarch64/NEON) ist geplant.

Bei der Implementierung der Komprimierungsalgorithmen entstand die Idee für eine angepasste Methode mit höherem Komprimierungspotential. Die Umsetzung dieser Idee ist ein weiterer Meilenstein.

Im Rahmen der *Prototyp-Entwicklung* lag der Fokus auf forensischen Untersuchungen eines bestehenden Datensatzes. Dies ist besonders hilfreich, um bei der Entwicklung stets mit deterministischen Datenströmen arbeiten zu können. Als wichtige Erweiterung von `NYGMA/RIOT` steht die *Live-Datenverarbeitung* an oberster Stelle.

Mit Hilfe des *Live-Capturing* wird eine Lösung zum Domain-Name-System (DNS) Monitoring entwickelt, die basierend auf den Eigenschaften von `NYGMA/RIOT` als leichtgewichtiges Intrusion-Detection-System (IDS) eingesetzt werden kann.

Die Einarbeitug in die notwendigen Technologien während der Projektphase war mit erheblichen Lernkurven verbunden. Der Einsatz von Prozessor spezifischen SIMD Befehlen erfordert besondere Sorgfalt bei der Speicherverwaltung bzw des Speicherlayouts für die Komprimierungsverfahren. Leider waren viele Open-Source Projekte schlecht dokumentiert und die Implementierungen nicht aussreichend um Sie ohne "Reibungsverluste" zu verwenden. Viel Arbeit war notwendig umd diese Bibliotheken einsetzen zu können. Auch die Wahl der Datenstruktur zum Erstellen des Indexes gestaltete sich aufwendig. Es wurden unterschiedliche `B+Tree` Implementierungen aus der Datenbanktechnik evaluiert und die Performanz der unter der für dieses Projekt zu erwartenden Work-Load abzuschätzen. Denn die Implementieurngen aus der `C++` Standard-Template-Library (STL) waren immer langsamer und benötigten mehr Speicher als die Alternativen. Dies ist insbesondere für die nächste Ausbaustufe der Live-Analyse wichtig. Der Benutzer kann die Implementierung durch Templates in jedem Fall frei wählen und so dem Anwedungsfall besser anpassen.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Wie bereits anfänglich erwähnt wurde darauf verzichtet ein spezielles Format für die Indizierung der Roh-Daten einzuführen. Der Hintergrund dieser Entscheidung war, dass durch die Verbreitung von selbstkomprimierenden Dateisystemen wie `ZFS` es eigentlich nicht mehr notwendig ist ein neues Format zu entwickeln mit dem Hauptziel die Daten zu komprimieren. Bei bereits existierenden Datenbestäden kann man unter Umständen die Daten nicht einfach ein 2tes mal abspeichern bzw konvertieren da die Datenmenge an sich zu groß hierfür wäre denn zumindest kurzfristig würde ja die doppelte Speicherkapazität benötigt.

Eine Implementierung der Live-Erfassung (Capture) wurde ebenfalls aussen vor gelassen da hierfür nicht nur die Erfassungskomponente implementiert werden müsste sondern auch Betriebssystem und hardwareabhängige Faktoren beachtet werden müssten. Im Rahmen der Projektdauer wäre das schlicht weg nicht möglich gewesen.

Anstatt halbfertige Speicherformate und Erfassungs-Module zu entwickeln wurde der Fokus auf Test-Driven-Development gesetzt. So werden für jede Softwarekomponente Unit-Tests zur Verfügung gestellt. Weiterhin wurden Fuzzing-Module implementiert, die die einzelnen Softwarekomponenten einem automatiserten

und randomisierten Stresstest unterziehen können. Durch deren Einsatz schon frühzeitig während der Entwicklung konnten abweichende Testergebnisse ausgewertet und Fehler beseitigt werden.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Das Projekt kann über GitHub bezogen werden. Pull-Requests und der Issue-Tracker sind aktiviert. Das Projekt beinhaltet eine kurze Dokumentation sowie Demonstration wie es benutzt werden kann. Demonstationsvideos und ein Beispiel Workflow kann über die Prototypefund-Demoweek Website erreicht werden. Der Link zur Prototypefund Seite wird noch in die Dokumentation mit aufgenommen.

- Projekt-Seite: https://github.com/stackless-goto/nygma
- Demoweek-Präsentation: https://demoweek.prototypefund.de/projects/17-nygma.html

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Die initial aufgestellte Kostenplanung konnte eingehalten werden. Bei der Projektplanung und Gestaltung der Meilensteine wurde diese durch wöchentliche Sprints bzw Sprint-Planung kontinuierlich überprüft. Kleinere Anpassungen bei der Zeitplanung konnten so frühzeitig erkannt und integriert werden. Größere Umplanungen oder Anpassungen waren nicht notwendig.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Während der Entwicklung wurden Schwachstellen wie `CVE-2020-1350` (Windows DNS Server) und `CVE-2020-1472` veröffentlicht. Hierbei kann durch manipulierte Netzwerkpackete die Kontrolle von betroffenen Systemen übernommen werden. Schwachstellen wie diese sind eine der Anwendungsgebiete bei denen `NYGMA/RIOT` seine Stärken als flexibles Werkzeug zur forensichen Analyse im Incident-Response Fall zeigen kann.

Dem Autor ist keine Open-Source Software bekannt die dem Use-Case und der Mächtigkeit von `NYGMA/RIOT` enstprechen.