Technische Universität Dresden

Fakultät Informatik

Professur für Datenschutz und Datensicherheit

Projekt HoneySens

11. Zwischenbericht

Forschungsprojekt zur Entwicklung eines Sensornetzwerks, um Angriffe auf die IT-Infrastruktur des Freistaates zu erkennen

Stand: 27.11.2018

Autor: Dipl.-Inf. Pascal Brückner
Betreuung: Dr.-Ing. Stefan Köpsell

1 Einleitung

Der Beauftragte für Informationssicherheit des Landes Sachsen prüft zur Verbesserung der IT-Sicherheit des Sächsischen Verwaltungsnetzes (SVN) die Grundlagen zum Aufbau eines Sensornetzwerkes, mit dem verdächtige Zugriffe auf Netzwerkdienste und -geräte erkannt und zuständige Stellen zeitnah informiert werden können. Das geplante System soll sich insbesondere durch eine leichte Bedienbarkeit und gute Skalierbarkeit auszeichnen. Hauptaugenmerk liegt dabei auf den Funktionalitäten, die die Wartung und Verwaltung der Sensorinfrastruktur betreffen. Weitere Faktoren, darunter die möglichst einfache und transparente Integration der zu entwerfenden Architektur in das bestehende Netzwerk, sowie ein autonomer, von anderen Netzwerkkomponenten und -diensten unabhängiger Betrieb sind ebenfalls zu berücksichtigen.

Der vergangene Projektzeitraum wurde in Zusammenarbeit mit der T-Systems MMS genutzt, um Erfahrungen im Praxisbetrieb mit der Version 1.0 zu sammeln und aufgetretene Probleme sowie an uns herangetragene Änderungswünsche zu verarbeiten. Während der Projektphase führte dies zu gleich zwei neuen Releases. Version 1.0.1 beinhaltet alle Änderungen aus dem 10. Zwischenbericht, die nicht mehr in die Version 1.0.0 eingeflossen waren, darunter der vereinfachte Entwicklungsprozess mit Hilfe des auf Docker basierenden Entwicklungsservers und eine erweiterte Ansicht des aktuellen Sensor-Status für Nutzer mit der rolle Beobachter sowie zahlreiche kleinere Bugfixes und Detailverbesserungen, insbesondere beim Layout der Web-Oberfläche. Zudem unterstützt diese Version die automatisierte Aktualisierung von Sensor-Zertifikaten, was in einem separaten Kapitel erläutert werden wird. Version 1.0.2 baut auf den Zertifikats-Updates auf und erlaubt es, die Zertifikate der gesamten internen PKI bei Bedarf zu verlängern, was Beispiel beim Ablauf der bereits ausgestellten Zertifikate (typischerweise mit einer Gültigkeit von einem Jahr) notwendig wird. Da er Produktivbetrieb in verschiedenen Netzen bereits seit einer Weile anhält, wurden diese Maßnahmen unumgänglich. Weitere Neuerungen in diesem Release betreffen die Sicherheit des Servers - hier wurden auf Basis eines internen Penetrationstests der MMS diverse HTTP-Header und -Parameter angepasst - und auch die Usability: So ist es nun möglich, den SMTP-Port für ausgehende Mails benutzerdefiniert festzulegen. Auch das war eine Anforderung aus dem praktischen Betrieb, wo der von HoneySens standardmäßig genutzte alternative SMTP-Port 587 oft nicht verfügbar war.

Weiterhin lag der Fokus während des gesamten Projektzeitraumes auf der Verbesserung der Sensor-Interaktivität. So sind nun die Honeypot-Dienste Conpot und Dionaea für die Nutzung auf allen unterstützten Plattformen (wieder) verfügbar, Glastopf hat zudem rudimentäre Unterstützung für HTTPS erhalten und Cowrie wurde einem Update unterzogen. Details zu diesen Vorgängen folgen im entsprechen Abschnitt.

Die im Ausblick des letzten Projektberichtes angesprochene Analyse der Ereignisdaten aus dem TU-Netz wurde zudem fortgeführt, ist aber noch nicht abgeschlossen. Einige erste Resultate aus diesem Prozess - im Wesentlichen eine Bestandsaufnahme der vorhandenen Daten - sind an diesen Bericht angehangen und knapp zusammengefasst.

Die im ersten Zwischenbericht beschriebenen Anforderungen behalten ihre Gültigkeit und gelten weiterhin als Richtlinie für zukünftige Entwicklungen. Sie werden an dieser Stelle nicht erneut ausgeführt.

2 Sachstand

Dieser Abschnitt gibt einen Überblick über die vom 01. Juli 2018 bis zum 30. November 2018 verzeichneten Fortschritte im HoneySens-Projekt. Im ersten Abschnitt wird der Prozess beim Entwurf der neuen Zertifikatsverlängerungs-Strategie detailliert ausgeführt. Dem folgt ein Überblick über die neuen und aktualisierten Sensordienste sowie im letzten Abschnitt ein kurzer Blick auf den (anonymisierten) Datensatz der Forschungsinstallation am Netz TU Dresden. Teilbereiche, die in den Anforderungen des ersten Projektberichtes genannt, aber hier nicht weiter ausgeführt werden, sind Gegenstand der zukünftigen Weiterentwicklung.

2.1 PKI: Automatische Aktualisierung ungültiger Zertifikate

Eine Reihe von HoneySens-Installation befinden sich nun schon seit geraumer Zeit im aktiven Betrieb, darunter an der TU Dresden und im Sächsischen Verwaltungsnetz. Die Kommunikation in einer solchen Infrastruktur ist asymmetrisch verschlüsselt: Eine zentrale, vom Server verwaltete *Certificate Authority* generiert und signiert Zertifikate für alle Sensoren und auch das standardmäßig generierte TLS-Zertifikat für den Zugriff vom Browser des Administrators wird von dieser CA ausgestellt. Die Gültigkeit solcher Zertifikate ist jedoch naturgemäß begrenzt, konkret stellt HoneySens standardmäßig Zertifikate mit einer Laufzeit von einem Jahr aus. Nach deren Ablauf werden die Zertifikate ungültig und sollten zurückgewiesen werden - das tatsächliche Verhalten hängt jedoch von der jeweiligen Implementierung und Konfiguration ab. Zusätzlich wurde das standardmäßig verwendete Hash-Verfahren für das Signieren in neueren OpenSSL-Versionen vom inzwischen als unsicher eingestuften Verfahren *MD5* auf *SHA256* geändert. Der HoneySens-Server nutzte jedoch im Auslieferungszustand weiterhin MD5, was letztendlich bei einem Sicherheitsaudit durch das Land Sachsen bemängelt wurde. Somit kam zusätzlich zur Anforderung, abgelaufene Zertifikate mit möglichst geringem Mehraufwand aktualisieren zu können, noch der Wunsch hinzu, bei dieser Gelegenheit auch ein zeitgemäßes Hash-Verfahren einzusetzen.

Praktisch führte dies zur Konzeption eines Updateverfahrens sowohl für das interne CA-Zertifikate als auch alle davon unterschriebenen Sensor- und TLS-Zertifikate. Es obliegt nun dem Nutzer, diesen Prozess in einer laufenden Instanz über das Web-Interface des Servers zu initiieren. Daraufhin werden im Hintergrund folgende Schritte durchgeführt:

1. Erzeugung eines neuen, gültigen CA-Zertifikats auf Basis des vorhandenen privaten CA-Schlüssels.
2. Erzeugung eines neuen, mittels des neuen CA-Zertifikats unterschriebenen TLS-Zertifikats (nur für Installationen relevant, die kein externes TLS-Zertifikat einbinden).
3. Erzeugung neuer Sensor-Zertifikate auf Basis der bestehenden privaten Schlüssel für alle registrierten Sensoren.
4. Neustart betroffener Server-Dienste (dies erfolgt vollautomatisch innerhalb des Containers)

Alle neu erzeugten und aktualisierten Zertifikate nutzen nun automatisch das Hash-Verfahren SHA256.

Nachdem diese Schritte abgeschlossen sind, ist der Server mit den neuen Zertifikaten wieder online und es obliegt nun den Sensoren, ihre lokale Konfiguration entsprechend anzupassen. Zu diesem Zweck wurde der Sensor-Manager - ein Prozess, der auf jedem Sensor die Kommunikations- und Verwaltungsschnittstelle mit dem Server darstellt - dahingehend angepasst, dass nun auch auf Basis einer bereits erfolgreich

etablierten und authentifizierten Verbindung der Server die Möglichkeit hat, dem Sensor neue Zertifikatsdaten bereitzustellen. Hierfür mischt der Sensor jeder regelmäßigen Polling-Anfrage den Fingerprint des TLS- und seines eigenen Zertifikates bei. Wenn der Server dann eine Abweichung feststellt, etwa weil inzwischen neue Zertifikate mit neuen Fingerprints generiert wurden, sendet der Server das komplette Zertifikat über die gesicherte Verbindung zum Manager, der es anschließend innerhalb der Sensorkonfiguration aktualisiert. Zukünftige Anfragen führt der Sensor dann mittels der neuen Zertifikate durch. Dieses Verfahren kann zudem auch genutzt werden, um extern eingebundene TLS-Zertifikate automatisch zu aktualisieren. Der Vorteil dieser Methode ist, dass Administratoren nicht händisch Zertifikate auf den Sensoren aktualisieren müssen. Lediglich ein Firmware-Update wird notwendig, was aber ebenfalls ein an früherer Stelle beschriebener Automatismus ist.

Der Entwurf und die Implementierung dieses Verfahrens durchliefen einige Schritte bis zu dieser finalen Lösung, die zudem nur Zertifikate verlängert, aus Sicherheitsgründen aber derzeit keine privaten Schlüssel verändert. Zunächst wurde eine aus zwei Phasen bestehende Lösung implementiert, bei der im ersten Schritt zunächst neue Zertifikate erzeugt und temporär als „zukünftige“ Zertifikate abgelegt werden. Danach musste gewartet werden, bis alle Sensoren einmal den Server kontaktiert und ihre lokale Konfiguration aktualisiert haben, woraufhin vom Nutzer der zweite Schritt manuell über das Web-Interface angestoßen wurde: Das Überschreiben der alten mit den neuen Zertifikaten und der Neustart der Serverdienste. Es stellte sich bei Versuchen schließlich heraus, dass sich dieser komplexe Vorgang auf einen Schritt ganz im Sinne der Benutzerfreundlichkeit reduzieren lässt. Die finale Nutzerschnittstelle im Web-Frontend, über das Administratoren diesen Vorgang anstoßen können, ist in Abbildung 1 dargestellt.

Abbildung 1: Benutzerschnittstelle für die Zertifikatsverlängerung

2.2 Sensor-Interaktivität: Neue und aktualisierte Honeypot-Dienste

Die Hauptaufgabe des HoneySens-Systems ist die Bereitstellung von Honeypot-Software auf den virtuellen und physischen Sensoren. Die hierdurch gesammelten Daten stellen den Mehrwert dieser Lösung dar. Aus diesem Grunde wird bei der Weiterentwicklung des Systems immer auch darauf geachtet, das Arsenal an verfügbaren Diensten schrittweise auszubauen und an die Anforderungen der Nutzer anzupassen, sowie bereits vorhandene Dienste auf möglicherweise neue Revisionen zu aktualisieren. Auch in dieser Projektphase war das der Fall, in der Folge werden die betreffenden Dienste und die jeweiligen Neuerungen kurz zusammengefasst.

Conpot : Ein wichtiger Neuzugang bei den verfügbaren Honeypot-Diensten ist Conpot[1], ein Open-Source ICS/SCADA-Honeypot, der Industriesteuerungen simulieren kann und hierfür eine Reihe von entsprechenden TCP/IP-basierten Protokollen spricht. Eine schnelle Konvertierung dieses Honeypots gab es durch die MMS zu Demozwecken schon seit geraumer Zeit, für die alltägliche Nutzung wurde jedoch die Modifikation der aktuellen Version in das für HoneySens nötige Dienst-Format von Grund auf neu begonnen. Das resultierende Sensor-Paket antwortet auf den TCP-Ports 80 (*HTTP*), 102 (*S7Comm*), 502 (*Modbus*) sowie dem UDP-Port 47808 (*Bacnet*) und leitet Information über das angesprochene Protokoll und den anfragenden Client wie gewohnt als Ereignis an den Server weiter.

Dionaea ist ein bekannter Gast in der Liste der verfügbaren Sensor-Dienste, u.a. zur Simulation von Windows-Dateifreigaben über das SMB-Protokoll. Der ursprüngliche HoneySens-Prototyp hatte eine alte Version dieses Honeypots bereits mit an Board. In der Zwischenzeit hat jedoch ein neues Team die Weiterentwicklung des Honeypots vom ursprünglichen Autor übernommen[2] und dabei weite Teile des Build-Systems und der Modulstruktur des Projektes modifiziert, weshalb das damalige HoneySens-Reporting-Modul für Dionaea nicht mehr kompatibel war. Während dieser Projektphase wurde Dionaea erneut aufgegriffen und ein neues Logging-Modul geschrieben, das mit aktuellen Versionen des Honeypots und des HoneySens-Servers zusammenarbeitet. Weiterhin wurde Dionaea, wie alle anderen Honeypot-Dienste auch, in das Container-Format überführt und während dieses Prozesses darauf geachtet, dass das entstehende Docker-Image möglichst klein zu halten. Dionaea bringt viele Abhängigkeiten mit sich, was aber im Zusammenhang mit dem begrenzten Speicherplatz auf physischen Sensoren ungünstig ist, wenn viele Dienste parallel betrieben werden sollen. Das resultierende Archiv bewegt sich nun mit einer Größe von ca. 140 MB in einem ähnlichen Rahmen wie die anderen Dienste.

Glastopf ist ein beliebter HTTP-Honeypot, der bereits seit längerem Teil der HoneySens-Serviceliste ist. Bisher war die Funktionalität jedoch vollständig auf die Bereitstellung von Pseudo-Webseiten über das HTTP-Protokoll beschränkt. Eine verschlüsselte Verbindung mit dem Honeypot mittels HTTPS war jedoch bisher nicht möglich. TLS-Support ist in der Anwendung selbst zwar vorgesehen, deren Implementierung hat jedoch seit Jahren keine Aktualisierung erfahren und unterstützt somit keine aktuellen Cipher-Suiten, wie sie moderne Browser voraussetzen. Aus diesem Grunde wurde bei der Implementierung auf das Tool *stunnel*[3] gesetzt, einem kleinen TLS-Proxy, der vor die eigentliche Anwendung geschalten wird. Somit bietet jeder Sensor, auf dem Glastopf betrieben wird, eine gültige TLS-Schnittstelle an. Ein zum jetzigen Zeitpunkt noch ungelöstes Problem ist allerdings, dass die dadurch erzeugten Ereignisse ihre Quellinformation verlieren und somit unter Umständen (abhängig davon, ob zur selben Zeit auch noch andere Ports kontaktiert werden) nicht eindeutig einem Absender zugeordnet werden können. An Lösungen für diese Herausforderung wird bereits gearbeitet, denkbar ist hier beispielsweise ein zusätzliches Modul für den Sensor-Manager, das alle ein- und ausgehenden Flows des Sensors auswertet und somit auch in der Lage ist, von verschiedenen Diensten eines einzelnen Sensore generierte Ereignisse noch lokal zusammenzuführen.

[1] `http://conpot.org/`
[2] `https://github.com/DinoTools/dionaea`
[3] `https://stunnel.org/`

Cowrie : Dieser vielseitige SSH-Honeypot ist ein aktiv gepflegtes OpenSource-Projekt[4], in das beständig Neuerungen und Fehlerbehebungen aus der Community einfließen. Während Tests der Honeypot-Dienste stellte sich heraus, dass neu gebaute Cowrie-Dienste nicht mehr antworteten, obwohl sich an den Quellen im HoneySens-Repository seit einiger Zeit nichts verändert hatte. Bei weiteren Nachforschungen stellte sich heraus, dass die für HoneySens gepinnte Cowrie-Version 1.2.0 nicht mehr mit neuartigen Versionen der Twisted-Bibliothek, von der dieser Honeypot abhängig ist, zusammenarbeitet. Eine Aktualisierung der betreffenden Komponenten schuf hier Abhilfe und ist nur ein Beispiel dafür, wie bestehende Honeypot-Dienste auch weiterhin gepflegt werden.

2.3 Ereignisdaten an der TU Dresden

Seit einigen Jahren wird an der TUD eine HoneySens-Installation zu Forschungszwecken betrieben. Diese Installation dient im Wesentlichen dem Test neuer Features, eine anonymisierte Auswertung der Ereignisdaten ist jedoch auch vorgesehen und wurde im Rahmen dieser Projektphase begonnen. Ziel der Analyse ist, den Nutzen der Honeypot-Umgebung anhand des Forschungsnetzes beispielhaft zu evaluieren und zudem effiziente Methoden zur Auswertung derartiger Ereignisdaten zu finden. An dieser Stelle soll ein kurzer Überblick über den Datensatz gegeben werden, eine detaillierte Interpretation der Daten ist für die Zukunft vorgesehen.

Der vorliegende Datensatz wurde dem laufenden System am 30. Mai 2018 entnommen und beinhaltet Ereignisse rückwirkend bis zum 4. Februar 2016. Insgesamt handelt es sich um 26823 aufgezeichnete Ereignisse nebst zusätzlichen Ereignisdaten wie beispielsweise empfangenen Paketen oder von Honeypots generierten Metadaten. Einen Überblick über die Struktur des Datensatzes gibt Abbildung 2. Jede der drei roten Kästen bezeichnet eine Tabelle in der Datenbank, die daraus extrahierten Daten liegen uns vollständig im CSV-Format vor.

Abbildung 3 gibt einen Überblick über die Anzahl der aufgezeichneten Ereignisse geordnet nach Sensor und Kategorie. Alle Sensoren befinden sich innerhalb verschiedener Mitarbeiter-, Studenten- und Gästenetze der TU Dresden, eine genauere Aufschlüsselung der Standorte und die Zuweisung zur gezeigten Graphik ist zum Schutz der Betreiber jedoch nicht möglich. Die Klassifikation der gezeigten Ereignisse kann der vorangegangenen Strukturgraphik entnommen werden: „2“ kennzeichnet Verbindungsversuche, also letztlich Ereignisse, die vom Recon-Catch-All Dienst der Sensoren aufgezeichnet wurden. Die Klassifikation „3“ steht hingegen für alle Ereignisse, die direkt von Honeypot-Software wie beispielsweise Cowrie erzeugt wurden. Vorfälle der Kategorie „4“ wurden vom Recon-Dienst als „Scanversuch“ gemeldet und bezeichnen zusammenfasste Verbindungsversuche, die binnen weniger Sekunden von einer einzelnen Quelle wahrgenommen wurden. Es fällt sofort auf, dass die Gesamtzahl der Ereignisse sehr ungleichmäßig auf die Sensoren verteilt ist, was wie erwartet stark von den Netzen, in denen die Geräte platziert wurden, abhängig ist. Nebenstehende Tabelle schlüsselt zudem die Ereignisse zusätzlich nach den zehn am häufigsten kontaktierten Ziel-Ports auf. Hier führt Port 137 als Vertreter für das NetBIOS-

Port (TCP/UDP)	# Ereignisse
137	33381
80	19609
8611	4356
161	4114
9	1769
22	1487
40000	1354
2054	1335
445	1083
12287	1070

[4] `https://github.com/cowrie/cowrie`

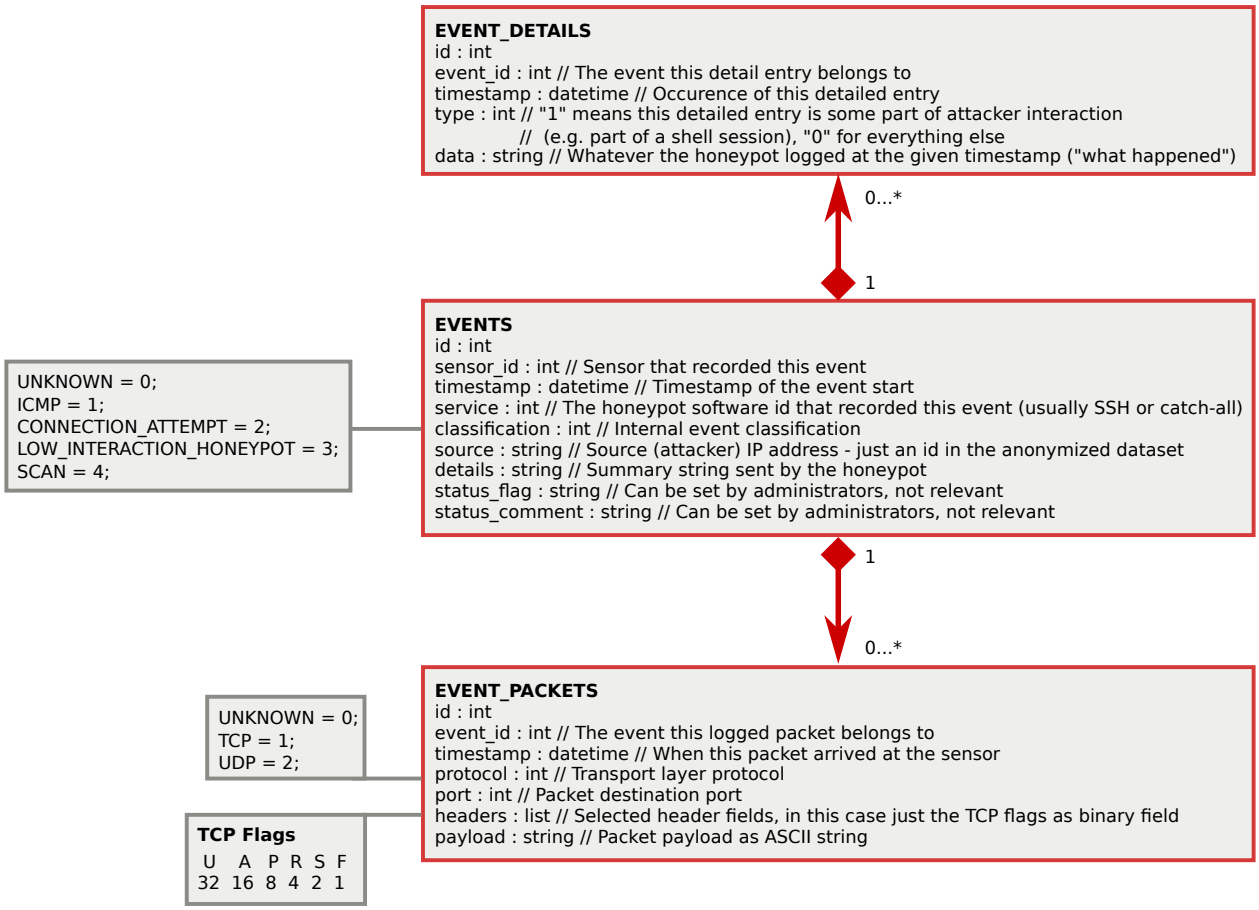

Abbildung 2: Tabellenstruktur des TU-Ereignisdatensatzes

Protokoll die Liste an, gefolgt von TCP-Port 80 (HTTP). Bereits an dritter Stelle folgt mit Port 8611 eine Position außerhalb der „Well known ports“, die nicht mehr eindeutig einem Protokoll zugewiesen werden kann ohne die übertragenen Daten genauer zu betrachten.

Eine feingranulare Analyse der Ereignisse steht derzeit noch aus. Hier ist insbesondere beabsichtigt, geeignete Verfahren zur möglichst automatisierten Analyse von Honeypot-Datensätzen zu finden. In der einschlägigen Literatur gibt es hierzu bereits eine Reihe von Veröffentlichungen, der Fokus liegt hier jedoch meistens nur auf Honeypots, die mit öffentlichen IP-Adressen betrieben werden. HoneySens mit seinem Fokus auf das Innere von Netzen hat hier jedoch andere Voraussetzungen, weshalb viele der vorgeschlagenen Verfahrenen nicht anwendbar sind. Nach einer genaueren Untersuchung einiger Beispieldaten aus dem Datensatz können jedoch schon jetzt zwei Schlussfolgerungen gezogen werden:

1. Die Mehrheit der aufgezeichneten Vorfälle stellen keine kritischen Sicherheitsvorfälle dar, sondern sind wiederkehrende Ereignisse, die von den Administratoren nicht auf die Whitelist gesetzt wurden. Nicht alle Administratoren behalten ihre Sensoren tatsächlich im Auge, untersuchen Vorfälle und pflegen die Whitelist. In der Folge kann es zu einer Explosion von Ereignissen kommen, die die spätere Auswertung massiv erschweren. Auf die Bedeutung und den Pflegebedarf der ersten Wochen nach Inbetriebnahme eines Sensors wurden zwar alle Teilnehmer hingewiesen, es muss

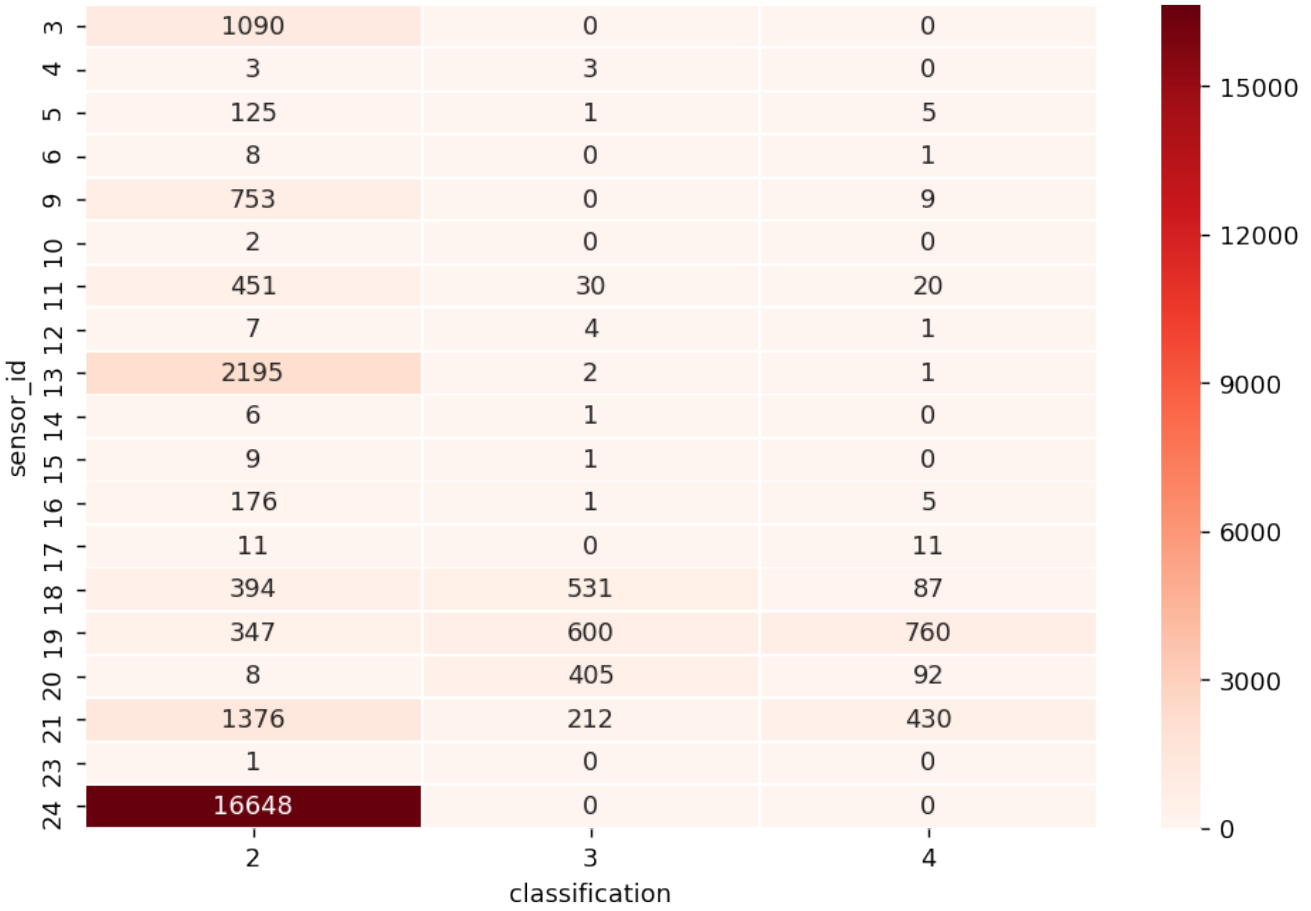

Abbildung 3: Ereignismengen aufgeschlüsselt nach Sensor und Klassifikation

aber damit gerechnet werden, dass diese „Lernphase“ entfällt. In diesem Fall bietet das System derzeit keine Hilfsmittel, um die Flut an Ereignissen leichter verarbeiten zu können. Dies ist ein offener Punkt für die zukünftige Weiterentwicklung.

2. Zahlreiche Ereignisse im Datensatz liefern starke Indikatoren dafür, tatsächlich Sicherheitsvorfälle zu sein. Dies allerdings mit der Einschränkung, dass es sich hierbei auch um Tests der zuständigen Administratoren handeln kann, um die Funktionsweise der Sensoren zu testen.

In welchem Verhältnis die False-Positives zu interessanten Vorfällen stehen und wie HoneySens Nutzer bei dieser Unterscheidung in Zukunft assistieren könnte, wird Gegenstand weiterer Untersuchungen sein.

3 Ausblick

Die nächste Projektphase hat neben den üblichen Pflegearbeiten am System und der Berücksichtigung von Nutzerinteressen primär das Ziel, das Fingerprinting der Honeypots zu erschweren. Hier sind einfache Modifikationen der Standardkonfiguration der Sensordienste denkbar, aber auch komplexere Anpassungen wie beispielsweise das Modifizieren des Antwortverhaltens der Sensoren auf niedrigen Protokollebenen können helfen, die Erkennung der Sensoren für Angreifer zu erschweren. Hier sollen zunächst verwandte Arbeiten auf dem Gebiet des Honeypot-Fingerprinting untersucht und im Anschluss eine Reihe von möglichen Lösungsmöglichkeiten diskutiert und bei Bedarf implementiert werden.

Zusätzlich wird die Auswertung der Ereignisdaten aus dem Datensatz der TU Dresden fortgesetzt. Ziel ist hier, sowohl mehr Erkenntnisse über die Sinnhaftigkeit einer Honeypot-Installation in einem Forschungsnetz als auch Ansätze zur Optimierung sowohl des Honeypot-Betriebes als auch der Auswertung der davon erzeugten Daten zu gewinnen.

Weiter in die Zukunft gedacht, ergeben sich für die Weiterführung des Forschungsprojektes weiterhin eine Reihe von interessanten Ansatzpunkten:

Sichtbarkeit : Die Betrachtung dieser Thematik aus wissenschaftlicher Sicht wurde bereits in Zwischenberichten erörtert. Als ein unmittelbarer Schritt wäre allerdings denkbar, Sensoren mit mehreren fixen IP-Adressen gleichzeitig auszustatten, um Administratoren ein Werkzeug zur Verfügung zu stellen, mit dem einzelne Sensoren mit statischen Mitteln für potentielle Angreifer prominenter platziert werden können. Diese Lösung könnte insbesondere für KMUs interessant sein, um mit nur wenigen Sensoren eine größere Bandbreite des Unternehmensnetzes abdecken zu können.

Firmware-Signatursystem : Um dem Fall vorzubeugen, dass ein erfolgreicher Angriff gegen den Server in manipulierter Firmware auf den Sensoren resultiert, ist ein Verfahren zu entwickeln, mit dem Firmware bei ihrer Erzeugung signiert und vor der Installation automatisch überprüft werden kann.

4 Anhang

Diese Auflistung soll die zuvor beschriebenen Prozesse und Veränderungen anhand der im Laufe des Forschungsprojektes in der Versionsverwaltung hinterlegten Kommentare für jede neue Softwarerevision dokumentieren.

Revision	Änderungen
557	HTTPS support for Glastopf
558	Conpot honeypot added to services
559	Dionaea honeypot added to services
560	- Dionaea event data clarified with textual descriptions - Dionaea image size reduced
561	Conpot: - Detailed data split into chunks because the server currently limits the length of such entries - Ports 161/UDP and 623/UDP removed from the supported protocols because those don't emit any responses or logs
562	- All compoenents: Support for sensor certificate update propagation added - Update script reissues all certificates as sha256 when updating to server 1.0.1 - New sensors certificates are now issued with a sha256 digest and no additional fields set next to CN - Fixed a bug in the setup modul which kept polling even after the update was done - Fixed a permission bug within the dev environment
563	- Server version set to 1.0.1 - The observer role can now view service assignments and the event status popup
564	gitignore file added
565	- MMS style update: Button hover color fixed - X-Frame-Options header set via apache
566	- MMS style update: Toggle button coloring
567	- Sidebar styling fixed on mobile (build number doesn't overlap with other content anymore) - MMS branding adjusted as well
569	- Missing glastopf files added - Server compose file now automatically restarts the server on boot
570	Fixed a bug in the dockerx86 firmware update procedure

571	Cowrie service fix: Install a specific twisted version, because cowrie 1.2.0 doesn't work with newer ones
572	Release Server Version 1.0.1 - Branding button colors adjusted - Release notes updated
573	- Fixed a bug that prevented the dev server from building correctly with newer grunt versions - Unbranded version fixed (button colors were broken due to branding style changes) - MMS style adjusted accordingly
574	Dev server build process overhaul: - Usage of docker compose - The registry is now available within the dev server - Fixed a bug that prevented a build with newer versions of Grunt
581	Function to overhaul the full internal certificate chain of a running system added. This generates a new CA cert and issues new certificates for both TLS and sensors. The sensor manager and firmware was adjusted accordingly.