Technische Universität Dresden

Fakultät Informatik

Professur für Datenschutz und Datensicherheit

Projekt HoneySens

Schlussbericht

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

Stand: 30.06.2019

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.

Dieser abschließende Bericht fasst die Ergebnisse des vergangenen Projektzeitraumes als auch des gesamten Forschungsprojektes zusammen. Es ist in mehrjähriger Konzeptions- und Entwicklungsarbeit gelungen, eine stabile und auch produktiv nutzbare Honeypot-Plattform zu entwickeln, die sowohl den besonderen Anforderungen des Freistaats Sachsen als auch den kleinen und mittelständischen Unternehmen gerecht wird und sich folglich dort auch im Einsatz befindet. Zusätzlich sind große Teile des Projektes unter einer freien Lizenz verfügbar und werden in einem öffentlichen Repository weiterhin gepflegt. Die T-Systems Multimedia Solutions GmbH deckt dabei auch in Zukunft den kommerziellen Aspekt des Projektes ab und ist neben der Weiterentwicklung gemäß den Bedürfnissen ihrer Kunden vor allem mit dem Betrieb der verschiedenen HoneySens-Installationen beschäftigt. Die Plattform ist dabei inzwischen sowohl als komplett unabhängig *On-Premises-Lösung* sowie auch als *Cloud-Service* innerhalb der *Open Telecom Cloud* erhältlich.

Im ersten Quartal des Jahres 2019 wurde zudem an das an der TU Dresden neu gegründete *TUD-Cert* damit beauftragt, die sich nach wie vor zu Forschungs- und Testzwecken in Betrieb befindliche HoneySens-Installation durch ein produktives Deployment abzulösen. Hierfür kann dank einer Kooperationsvereinbarung von allen beteiligten Parteien auf das Code-Repository der T-Systems MMS zugegriffen werden, wobei im Kontext des TUD-Cert erfolgte Weiterentwicklungen direkt an die MMS zurückfließen. Eine erste Anforderungsanalyse für die Abdeckung weiter Teile des Netzes der TU Dresden mit Honeypots wurde bereits durchgeführt und darauf basierend einige Konzepte für die Umsetzung erarbeitet. Auf diese wird an späterer Stelle in diesem Bericht kurz genauer eingegangen werden.

Der vergangene Projektzeitraum wurde in Zusammenarbeit mit der T-Systems MMS genutzt, um die HoneySens-Plattform mit den Releases 1.0.3 und 1.0.4 weiter zu stabilisieren als auch mit Version 2.0.0 um eine Reihe von bereits über längere Zeit in Arbeit befindlichen Neuerungen zu bereichern. So wurde beispielsweise das Handling der Honeypot-Dienste auf der Sensorseite massiv überarbeitet, womit der Status der individuellen Dienste eines jeden Sensors für den Nutzer sichtbar gemacht werden konnte. Weiterhin kamen speziell auf Wunsch des Freistaats Sachsen eine Reihe von Möglichkeiten zum Debugging der Sensor-Komponenten hinzu, was sowohl ein ausführlicheres und feingranularer filterbares Logging als auch einen Vor-Ort-Debugging-Mechanismus für die BeagleBone-Sensoren umfasst. Weiterhin wurde das Repository an verfügbaren Honeypot-Diensten überarbeitet: Die verschiedenen Dienste wurden hierbei auf die jeweilige Upstream-Version aktualisiert und bei dieser Gelegenheit umkonfiguriert, um ein einfaches Fingerprinting der Dienste durch Angreifer zu erschweren. Auch das Web-Frontend hat Überarbeitungen erfahren: Die Ereignisübersicht kann fortan mittels neuer Filter- und Suchfunktionen durchsucht werden und das Modul für die Formularvalidierung in der kompletten Anwendung wurde

aus lizenzrechtlichen Gründen durch eine freie Alternative ersetzt. Details folgen im entsprechenden Abschnitt.

Der Abschluss des Forschungsprojektes bedeutet keinesfalls das Ende des Projektes HoneySens. Wie zuvor bereits erwähnt zeichnen sich für die zukünftige Weiterentwicklung mehrere Parteien verantwortlich, die gemeinsam an der Code-Basis arbeiten. Auch der regelmäßige Rückfluss von kommerziellen Neuerungen in das OpenSource-Projekt ist sichergestellt, so dass sich hoffentlich in Zukunft eine freie Community rund um das Projekt bilden kann. Die Domain `honeysens.org` wurde hierfür reserviert und verweist zum aktuellen Zeitpunkt noch direkt auf das GitHub-Projekt[1]. Eine vollständige Website für das freie Projekt ist allerdings bereits in Arbeit. Auch aus Forschungssicht besteht weiterhin Interesse, speziell im Kontext der *Deception Networks*, die in den letzten Montaten wieder mehr Relevanz auf den einschlägigen Konferenzen erfahren haben. Insbesondere die Anpassung von HoneySens für spezifische Anwendungsfälle, beispielsweise im industriellen Sektor oder dem IoT-Bereich, wird in Zukunft und für mögliche weitere auf dem jetzigen Stand aufbauende Forschungsprojekte eine Option bleiben.

[1] `https://github.com/sylencecc/honeysens-ce`

2 Sachstand

Dieser Abschnitt gibt einen Überblick über die vom 30. November 2018 bis zum 30. Juni 2019 verzeichneten Fortschritte im HoneySens-Projekt. Zunächst werden die allgemeinen Prozesse und Veränderungen beschrieben, die zur kontinuerlichen Wartung und Verbesserung der Software insbesondere in Zusammenarbeit mit der MMS und deren Kunden etabliert wurden. Im Anschluss wird auf die zahlreichen Neuerungen der Sensorsoftware und insbesondere auch das nun implementierte Vor-Ort-Auditing von BeagleBone-Sensoren eingegangen. Das Kapitel rundet eine Zusammenfassung der Arbeiten für die neue Formularvalidierung ab, bei der die letzte verbleibende aus Lizenzsicht problematische Komponente durch eine freie Alternative ersetzt wurde.

2.1 Stabilität und Zuverlässigkeit

Aus den verschiedenen aktiven HoneySens-Installationen innerhalb des Sächsischen Verwaltungsnetzes (SVN), an der TU Dresden und auch in mehreren von der T-Systems MMS betreuten kleinen und mittelständischen Unternehmen (KMUs) erhalten wir regelmäßig Feedback und Verbesserungsvorschläge, die dann in die Weiterentwicklung des Produktes mit einfließen. Die seit November 2018 herausgegebenen Releases 1.0.3 und 1.0.4 beinhalten neben juristischen firmenspezifischen Anpassungen v.a. kleinere Korrekturen, die die Usability im Alltag verbessern sollen. Als Beispiel kann an dieser Stelle die Honeypot-Dienstekonfiguration genannt werden, die ein Teil der Sensorübersicht ist. In einer großen Matrix aus interaktiven Checkboxen kann im Web-Interface jeder Dienst für jeden Sensor an- oder abgeschaltet werden. In der Praxis hat sich allerdings herausgestellt, dass Dienste dabei gern einmal versehentlich modifiziert werden, weswegen nun die gesamte Matrix standardmäßig inaktiv ist und erst nach Klick auf einen dafür vorgesehenen Button global entsperrt wird. Die Einführung derartig kleiner aber gefragter Features hat die Benutzerfreundlichkeit des Systems insgesamt gemessen an den Rückmeldungen deutlich steigern können.

Weitere inkrementelle Änderungen auf Basis von Nutzer-Feedback umfassten in diesem Zusammenhang eine Überarbeitung des automatischen Mailversands. Seit der SMTP-Port des Ziel-Mailservers nun frei konfigurierbar ist (siehe 11. Zwischenbericht), werden die Mailfunktionen von HoneySens rege genutzt. Dabei stellte sich heraus, dass unter bestimmten Voraussetzungen Nachrichten an mehr Empfängergruppen gesendet wurden als dies konfiguriert wurde. Zudem wurden die eingerichteten Filter (konkret die Whitelist) beim versenden von Nachrichten nicht berücksichtigt, was sich darin äußerte, dass Mail-Benachrichtigungen für Ereignisse versendet wurden, die im Frontend gar nicht existierten. Beide Probleme konnten mittels eines Testsystems rekonstruiert und im Anschluss behoben werden.

Im gleichen Rahmen wurde - ebenfalls durch die Rückmeldungen von Anwendern inspiriert - auch das „Verpackungsformat“ der Server-Software aktualisiert. Das entsprechende Archiv beinhaltet nun neben zusätzlicher Dokumentation zu Installation und Updates insbesondere auch ein Composefile, mit dem der Server direkt in Betrieb genommen werden kann. Da sich Docker Compose[2] inzwischen als eine Art Quasi-Standard zum Deployment von Container-Services ohne separaten Orchestrator etabliert hat, kann nun besagtes Composefile als Grundlage für eine neue Installation verwendet werden. Es kann bei Bedarf modifiziert und an die lokalen Gegebenheiten der jeweiligen IT-Landschaft angepasst werden.

[2] `https://docs.docker.com/compose/overview/`

2.2 Überarbeitung der Sensorsoftware

Die zentrale Komponente eines jeden HoneySens-Sensors ist ein schlicht *manager* genannter Prozess, der bereits in früheren Zwischenberichten genauer vorgestellt wurde. Er ist prinzipiell plattformunabhängig entworfen, hat jedoch eine modulare Struktur und erlaubt das Nachladen von plattformspezifischen Komponenten. Eine wesentliche Aufgabe dieser Module ist der Firmware-Update-Mechanismus, der je nach Plattform stark variiert. Beim Aufsetzen und (automatischem) Aktualisieren eines containerbasierten Sensors half dem Nutzer bisher ein kleines Python-Skript (der *launcher*), das mit den entsprechenden Parametern einen neuen Sensor-Container samt Volumes und Netzwerkkonfiguration erstellte oder - via Cron-Job - auch ein vollständiges Update eines solchen Containers vornehmen konnte. Die überschaubare Dokumentation des Skriptes und auch die sonst eher unübliche Herangehensweise an die genannten Anforderungen führten letztendlich jedoch dazu, den Update-Mechanismus und damit auch das gesamte Installationsverfahren zu überdenken. Da serverseitig bereits seit einiger Zeit Docker Compose zum Einsatz kam und die Administratoren damit entsprechend vertraut sind, wurde im Rahmen der Entwicklungsphase von HoneySens Version 2.0 ein neues Konzept für die Handhabung von Sensor-Containern konzipiert und implementiert.

Das Resultat: Docker-Sensoren nutzen nun ebenfalls Docker Compose und können sich mittels einiger Tricks auch selbstständig aktualisieren, ohne dass auf dem Hostsystem dafür irgendwelche gesonderten Skripte benötigt werden. Im Hintergrund wird dies durch den geschickten Einsatz von Volumes bewerkstelligt, womit aus dem Container heraus der Docker-Daemon des Hostsystems kontaktiert werden kann. Während eines Systemupdates wird dann lediglich die vom Server bezogene Firmware beim Docker-Daemon des Hostsystems registriert, ein paar Compose-spezifische Umgebungsvariablen angepasst und im Anschluss mittels Docker Compose aus dem Container heraus ein weiterer Sensor-Container gestartet, der als Teil seines Initialisierungsprozesses die bestehende Sensor-Konfiguration übernimmt und den alten Container entfernt. Bestehende containerbasierte Sensoren müssen leider händisch auf das neue Verfahren umgezogen werden, die Installation von neuen Instanzen ist jedoch einfacher als zuvor: Es genügen ein paar kleinere Anpassungen im mit ausgelieferten Composefile an die lokalen Gegebenheiten (bspw. der Pfad zur Sensor-Konfiguration), das separate Launcher-Skript entfällt. Das neue Verfahren hat sich in der Praxis bereits als erstaunlich robust und zukunftsfähig erwiesen und ist somit direkt in die Version 2.0 eingeflossen.

Ein mit diesem neuen Deployment-Verfahren einhergehender Vorteil ist, dass nun analog zum Server auch eine Entwicklungsinstanz eines Sensors gestartet werden kann, was die Weiterentwicklung der Sensorsoftware massiv vereinfacht hat. Mittels *GNU make* genügt nun ein `make dev` im Stammverzeichnis des Sensors, woraufhin automatisch ein Docker-Image des Sensors gebaut und im Entwicklungsmodus gestartet wird. Änderungen an der Code-Basis, bspw. des Manager-Daemons, werden direkt im laufenden Sensor gespiegelt und können sofort erprobt werden.

Auf Basis dieser Vorarbeiten konnten weitere gefragte Features integriert und als Bestandteil der Revision 2.0 ausgeliefert werden. So wurde beispielsweise das Logging des Manager-Prozesses überarbeitet und nutzt nun eine Hierarchie von Logging-Instanzen, deren Ausgaben über ein frei wählbares Logging-Level gefiltert werden können. Das Hinzufügen von Farbcodes für die Ausgaben und eine Vielzahl neuer Debug- und Informationsmeldungen erleichtern seitdem sowohl die Weiterentwicklung der Sensorsoftware als auch das Debugging bei aufgetretenen Problemen.

Eine weitere erwähnenswerte Neuerung ist, dass Sensor-Dienste (also die in Containern betriebenen Ho-

neypots) nun in einem isolierten Docker-Netzwerk betrieben werden, dessen Netzbereich vom Sensor-Verwalter frei bestimmt werden kann - entweder global oder auch individuell pro Sensor (siehe Abb. 1). Hintergrund ist, dass sich bei manchen Nutzern zuvor der von den Sensoren intern genutzte Netzwerkbereich für die Honeypot-Services (bisher fix `172.17.0.0/16`) mit produktiven Netzbereichen in der Firmenlandschaft überschnitten hat. Das hatte zur Folge, dass Anfragen an die Honeypots nicht mehr korrekt verarbeitet werden konnten und die Sensoren dadurch inaktiv wirkten. Während dieses Problem in den betreffenden Fällen durch einige manuelle Eingriffe gelöst werden konnte, wurde mit Version 2.0 ein allgemeingültiger Mechanismus integriert, über den im Konfliktfall das interne Honeypot-Netz beliebig umadressiert werden kann. Die entsprechenden Einstellungen können wie üblich komfortabel über das Web-Interface vorgenommen werden.

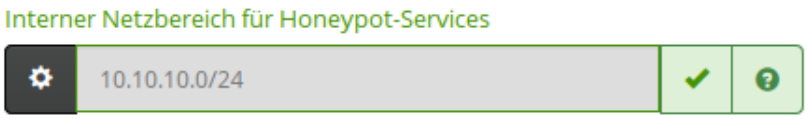

Abbildung 1: Konfiguration des Service-Netzwerkbereichs

Wie in der Einleitung bereits angedeutet, teilen Sensoren nun dem Server auch den Status der lokal laufenden Dienste mit. Hierfür wurden weite Teile der Dienste-Verwaltung im Sensor-Manager parallelisiert: Zuvor arbeitete der Sensor-Manager überwiegend seriell, womit beispielsweise der Download eines neuen Service-Images (für einen neu aktivierten Honeypot-Dienst) weitere Status-Updates vom Server nach hinten verschob, bis der entsprechende Download abgeschlossen und der Service gestartet war. Das hatte den unangenehmen Nebeneffekt, dass Sensoren während eines langwierigen Downloads vom Server fälschlicherweise als offline angezeigt wurden. Die Kernfunktionalität, nämlich das Verarbeiten von Honeypot-Ereignissen, war davon allerdings nicht betroffen, da diese Komponente seit jeher in einem separaten Thread betrieben wurde. Nichtsdestotrotz wurden nun alle weiteren Komponenten parallelisiert, wodurch sich die Rückmeldungen an die Nutzer bei der Umkonfiguration von Sensoren deutlich verbessert hat. Die Dienste-Matrix in der Sensorübersicht zeigt nun quasi als Nebeneffekt der vorigen Verbesserungen einen Live-Statusbericht der jeweiligen Dienste: farblich kodiert ist für Nutzer sofort ersichtlich, welche Dienste auf einem Sensor aktiv sind und welchen Status diese haben (grün zeigt einen betriebsbereiten Dienst, blau einen aktiven Installations- oder Updateprozess und rot einen Fehler an) - siehe Abb. 2.

Abbildung 2: Statusanzeige in der Sensor-Dienstematrix

Weitere Neuerungen der Sensorsoftware, die an dieser Stelle nur kurz erwähnt werden sollen, umfassen:

Cachen von Ereignissen : Falls der Server einmal nicht erreichbar ist, halten die Sensoren bis zu 1024 Ereignisse im Speicher vor und übermitteln diese, sobald die Verbindung wiederhergestellt ist.

Falls in diesem Zeitraum mehr Ereignisse eintreten, werden immer die letzten 1024 Ereignisse vorgehalten.

Netzwerkanbindung virtueller Sensoren : Bisher war es Aufgabe des zuständigen Administrators, Netzwerkverkehr an Sensor-Container weiterzuleiten. Im Zuge der oben genannten Umbauten verwaltet nun standardmäßig der Sensor-Daemon diesen Prozess und teilt sich einen Netzwerk-Stack mit dem Hostsystem. Somit nutzt der Sensor eines der Netzwerk-Interfaces des zugrundeliegenden Hosts, die dafür nötigen Firewall-Konfiguration nimmt der Sensor beim Start automatisch vor. Das alte Verhalten kann jedoch auf Wunsch aktiviert werden, falls die automatische Konfiguration fehlschlägt.

Dokumentation : Analog zum Server-Archiv beinhaltet nun auch die Sensor-Firmware alle nötigen Dateien für ein sofortiges Deployment.

2.3 BeagleBone-Auditing

Wenn in der Vergangenheit Probleme mit einem der physischen Sensoren auftraten, war zur Lokalisierung des Problems stets Handarbeit erforderlich: Das Gerät musste geöffnet und mittels eines USB-Kabels eine Verbindung mit einem anderen Rechner hergestellt werden. Anschließend konnten die Logs des Systems über eine lokale SSH-Session ausgewertet und bei Bedarf weitere Maßnahmen ergriffen werden. Mit steigender Anzahl an Sensoren wurde dieses Unterfangen schnell mühselig. Zwar neigen die BeagleBones selbst nicht zu Ausfällen, Bugs in der Software oder Netzwerkprobleme bei der Einbindung von neuen Sensoren traten jedoch durchaus auf und verursachten vergleichsweise viel Debugging-Aufwand.

Um diesem Problem in Zukunft besser begegnen zu können, wurde auf Anraten des SAX.CERT ein lokaler Audit-Mechanismus speziell für die Beaglebone-Plattform implementiert. Die Firmware analysiert dafür alle an das Gerät angeschlossenen USB-Geräte und sucht nach beschreibbaren Partitionen. Sobald eine solche gefunden wird (beispielsweise nach Anstecken eines USB-Sticks), legt der Sensor darauf ein Verzeichnis mit allen lokal vorhandenen Log-Daten ab. Diese können im Anschluss bequem an einem Rechner aufbereitet und auf mögliche Probleme untersucht werden. Um den Auditing-Prozess für den Benutzer transparent zu machen, zeigt der Sensor das Ende des Schreibvorganges zudem mit einem Farbcode an, wenn das entsprechende LED-Board installiert wurde.

2.4 Formularvalidierung

Ein weiterer Meilenstein auf dem Weg zum Release 2.0 war die Überarbeitung der Formularvalidierung im Web-Frontend. Diese ist dafür verantwortlich, Benutzereingaben mittels eines einfachen Regelwerks auf Plausibilität zu überprüfen, beispielsweise ob ein Feld, das eine Zeitangabe in Sekunden enthalten soll, auch tatsächlich mit einer Zahl gefüllt wurde. Noch aus der Zeit des Prototypen stammt die bisherige Implementierung mittels des ehemals freien JavaScript-Plugins *BootstrapValidator*. Das Projekt ist seitdem jedoch gewachsen und ist nach einer Namens- und Lizenzänderung (nun *FormValidation*[3]) nur noch gegen Gebühren kommerziell nutzbar.

[3] `https://formvalidation.io/`

Aus diesem Grund wurden parallel zu den anderen Arbeiten schrittweise alle Formulare auf eine alternative, freie Implementierung, nämlich den *bootstrap-validator* des Autors *1000hz*[4], umgestellt. Dies hatte zudem den positiven Nebeneffekt, dass dieses Plugin Validierungsregeln als HTML5-konforme Tags definiert und keine eigene Validierungssprache benötigt. Der resultierende Code ist somit leichter zu lesen, insbesondere da die Validierungsregeln meistens direkt innerhalb der Templates definiert werden können und keine separaten JavaScript-Regelsätze mehr benötigen. Der Umbau verschlang jedoch v.a. aufgrund der Komplexität einiger Formulare (insbesondere bei der Definition von Filtern) relativ viel Zeit, weswegen die neue Validierungsengine schlussendlich erst mit dem Release 2.0 fertiggestllt wurde. Es wurde Wert darauf gelegt, dass dieser Umbau für den Endanwender möglichst transparent ist und sich alle Formulare noch analog zur Vorgängerversion bedienen lassen. Zusätzlich konnten eine Reihe von Inkonsistenzen zwischen server- und clientseitiger Überprüfung behoben werden, was in manchen Situationen dazu führte, dass ein Formular im Browser als gültig markiert war und zugleich vom Server abgewiesen wurde.

[4] `https://1000hz.github.io/bootstrap-validator/`

3 Projektübersicht

Parallel zum HoneySens Release 2.0 im Sommer 2019 endet nach mehr als vier Jahren offiziell auch das zugehörige Forschungsprojekt, weshalb an dieser Stelle noch einmal ein Überblick über alle vergangenen Meilensteine gegeben werden soll. Die chronologische Aufstellung erhebt dabei keinen Anspruch auf Vollständigkeit.

Meilensteine 2015

Refactoring des Prototypen : Da das Forschungsprojekt aus dem Prototypen einer Diplomarbeit entstand, begann es mit einer massiven Überarbeitung der Codebasis, um eine langfristige Weiterentwicklung sicherzustellen. Dies beinhaltete die Auftrennung und Modularisierung weiter Teile des Codes sowie das Hinzufügen eines Build-Systems (*Grunt*) zum Zusammenfügen aller benötigten Komponenten. Im weiteren Projektverlauf wurde stets darauf geachtet, die Codebasis an allen Stellen modular zu halten. Eine spätere Einführung von Makefiles erleichterte die Entwicklung und den Bau der Software noch einmal weiter, damit auch Dritte ohne Einarbeitungsaufwand ein Testsystem aufsetzen können.

Mandantenfähigkeit : Die prototypische Implementierung unterstützte zwar mehrere Benutzer, allerdings keine Aufteilung derer oder auch anderer Ressourcen wie bspw. der Sensoren in eigenständige Gruppen. Da das Forschungsprojekt die Anpassung des Systems für die Gegebenheiten des SVN zum Inhalt hatte und einzelne Behörden und Organisationseinheiten unabhängig voneinander mit einem Server operieren sollten, war die Implementierung entsprechender Funktionen eine der ersten Maßnahmen. Die resultierende Architektur trennt Benutzer, Sensoren, Ereignisse, später auch Whitelist-Filter, Benachrichtigungsoptionen usw. rigoros gemäß der ihnen zugeordneten Gruppen.

Systemsicherheit : Beginnend mit der ersten Projektphase waren Sicherheitsaspekte der HoneySens-Plattform während des gesamten Projektes von zentraler Bedeutung. Das System soll Angreifer detektieren können, aber ihnen nicht selbst zum Opfer fallen. Aus diesem Grunde wurde frühzeitig eine robuste Architektur entworfen, deren Kommunikationswege verschlüsselt und alle Teilnehmer entsprechend autorisiert sind. So kommunizieren die Sensoren mit dem Server beispielsweise ausschließlich über verschlüsselte Verbindungen und authentifizieren sich mittels eines lokal vorliegenden Zertifikats.

Serverwartung : Die Notwendigkeit frei nutzbarer Wartungsfunktionen wie beispielsweise das Entfernen aller Ereignisse, das Zurücksetzen des Systems in den Auslieferungszustandes oder auch der benutzerfreundliche Umgang mit automatischen Updates wurde beim Einsatz von HoneySens im SVN schnell deutlich. Diese Funktionen wurden zwar nicht bei der alltäglichen Arbeit mit dem System benötigt, konnten aber gerade bei Funktionstests oder auch der Problembehandlung von großem Wert sein. Die Implementierung erfolgte schrittweise und gipfelte im voll automatischen Update-Assistenten, der die Datenbank und Datenverzeichnisse des Servers bei Bedarf aktualisiert.

Plattformunabhängigkeit der Sensoren : Der Prototyp unterstützte zunächst ausschließlich die BeagleBone-Plattform für Sensoren. Es wurde doch relativ schnell deutlich, dass sowohl für die Entwicklung als auch einen angestrebten Produktivbetrieb weitere Plattformen Bedeutung erlangen würden. Deshalb wurden schon zu Beginn in einem ersten Schritt Skripte hinzugefügt, die aus den Sensor-Quellen Debian-Pakete für verschiedene Architekturen erzeugten und auch auf handelsüblichen

`x64_64`-Systemen verwendet werden konnten. Die Einrichtung eines solchen Sensors war im Vergleich zu den inzwischen etablierten containerisierten Instanzen noch ziemlich umständlich und unterstützte auch keine atomaren Updates, erleichterte aber die Weiterentwicklung der Sensorsoftware enorm.

Whitelisting : Dass Low-Interaction-Honeypots je nach umgebendem Netz eine signifikante Menge an False-Positives erzeugen, wurde schon nach den ersten Einsätzen in produktiven Umgebungen deutlich. Damit diese quasi wertlosen Ereignisse den Administratoren nicht die Übersicht nahmen, wurde im Winter 2015 eine Filterfunktion integriert, bei der auf dem Server hinterlegte Regeln ein automatisches Whitelisting von „bekannten“ Vorfällen ermöglichen. In der Folge wurde auch bei Schulungen dieser Punkt deutlich hervorgehoben: Beim Einsatz der Honeypots ist nach Installation eine mehrwöchige Lernphase nötig, in der auftretende regelmäßige Ereignisse untersucht, bewertet und bei Bedarf auf die Whitelist gesetzt werden müssen. Andernfalls wird die Handhabung der kontinuerlich anwachsenden Ereignisliste massiv erschwert.

Recon-Service : Der Protoyp unterstützte zunächst nur eine begrenzte Auswahl an Honeypot-Projekten, darunter den damals noch *kippo* genannten SSH-Honeypot. Verbindungen von Angreifern zu TCP- oder UDP-Ports, für die kein Honeypot vorgesehen war, konnten zu diesem Zeitpunkt allerdings noch nicht aufgezeichnet oder ausgewertet werden. Aus diesem Grund wurde schließlich der *recon* getaufte Dienst entwickelt, der mittels eines lokalen Raw-Sockets jeglichen eingehenden Datenverkehr mitliest und auf alle nicht von anderen Honeypots beantwortete Anfragen antwortet. Dieser Dienst brachte HoneySens die Bezeichnung eines Frühwarnsystems ein, da somit alle eingehenden Datenpakete zumindest aufgezeichnet oder teils auch beantwortet werden können und ein vollständiges Monitoring einer Honeypot-IP-Adresse möglich wird.

Meilensteine 2016

Unterstützung von HTTP(S)-Proxy-Servern : Im Netz des SVN kommen an vielen Stellen Proxy-Server zum Einsatz, eine HTTP(S)-basierte Kommunikation ist zumeist nur über diese möglich. Aus diesem Anlass wurde die Sensor-Software frühzeitig mit vollständiger Unterstützung für verschiedene Arten von Proxy-Servern und diversen Authentifizierungsverfahren implementiert. Im Laufe der nachfolgenden Projektphasen wurde dieses Thema hin und wieder erneut relevant, da doch deutlich mehr Probleme als eingangs erwartet beim Zusammenspiel der Sensorkomponenten mit den Proxies auftraten. Insbesondere die an einigen Stellen eingesetzten Legacy-Proxies des *Microsoft Forefront Threat Management Gateway* bereiteten viele Probleme, wofür auch häufiges Degbuggen vor Ort an SVN-Knotenpunkten erforderlich war. Schlussendlich wurde ein lokaler Proxy-Dienst in die Sensorsoftware integriert, der die Kommunikation mit verschiedenen Typen von Proxy-Servern abstrahiert und sich inzwischen im Produktivbetrieb bewährt hat.

Sensorarchitektur : Vor der dritten Projektphase bestand die Sensorsoftware zunächst noch aus einer Reihe von losen Bash- und Python-Skripten, die schließlich in einige wenige Kernkomponenten aufgeteilt und mit klar definierten Interfaces versehen wurden. Die klar getrennten Zuständigkeiten erleichterten insbesondere die Unterstützung pattformspezifischer Features, da die entsprechenden Komponenten einfach ausgetauscht werden konnten. So brachte beispielsweise jede Plattform ihr eigenes Update-Skript mit sich.

Deployment mittels Docker : Viel Energie floss auch in die Erleichterung des Deployments neuer Server-Instanzen mittels Docker. Bevor dies später mit Docker Compose weiter vereinheitlicht wurde, war

noch die manuelle Interaktion mit Docker notwendig. Hierfür wurde das im Jahr 2016 weit verbreitete *Data-only Container* Pattern angewendet, bei dem ein separater Offline-Container alle Daten, bzw. Volumes eines produktiven Containers bindet, was für spätere Updates hilfreich ist. Mit der Weiterentwicklung der Docker-Plattform und neuen Features beim Umgang mit Volumes wurde dieser Ansatz später jedoch obsolet.

Beginn Testbetrieb : Sowohl im Netzwerk der TU-Dresden als auch innerhalb des SVN begann im Jahr 2016 der Testbetrieb mit einer wachsenden Anzahl von Sensoren. In beiden Fällen wurden Freiwillige gesucht und auch in großer Zahl gefunden, die Sensoren in ihren Struktureinheiten platzierten und somit wertvolle Daten und Erfahrungsberichte beisteuerten.

Interaktive Ereignisübersicht : Die ursprünglich einfache Liste von Ereignissen des Prototypen wurde bei einer größeren Anzahl sehr schnell unhandlich: Lange Ladezeiten und fehlende Filter- und Sortiermöglichkeiten erschwerten die Auswertung der Vorfälle. Nach einer vollständigen Überarbeitung des zugehörigen Frontend-Moduls lud die Liste nur noch angefragte Daten im Hintergrund bei Bedarf nach, was die Ladezeiten auf ein Minimum reduzierte, ließ sich nach verschiedenen Kriterien filtern oder sortieren und passte sich automatisch der Fenstergröße des Browserfensters an. In späteren Projektphasen wurde diese Ansicht schrittweise ausgebaut und um weitere Features wie dem gleichzeitigen Bearbeiten oder Löschen von einer Vielzahl von Ereignissen erweitert.

Penetrationstest : Um das Sicherheitskonzept von HoneySens auf die Probe zu stellen, wurde im Jahr 2016 ein Student im Rahmen seiner Bachelorarbeit damit beauftragt, einen umfangreichen Grey-Box-Penetrationstest durchzuführen. Hierfür analysierte er zunächst alle Schnittstellen und Kommunikationsstrukturen des Systems, erarbeitete einen Testplan und führte anschließend mittels einschlägiger Werkzeuge die Tests gegen ein mit dem damaligen Softwarestand bereitgestelltes System aus. Die Ergebnisse der Tests und resultierende Empfehlungen wurden später schrittweise in das Projekt mit eingearbeitet. Weitere Penetrationstest wurden später durch die T-Systems MMS durchgeführt und ebenfalls bei der Weiterentwicklung berücksichtigt.

Setup-Prozess für Installation und Updates : Um Nutzern die Installation und auch Aktualisierung des HoneySens-Servers über eine schriftliche Bedienungsanleitung hinaus zu vereinfachen, wurde ein Setup-Assistent implementiert, der selbstständig aktiv wird, wenn zusätzliche Informationen vom Nutzer benötigt werden. Dieser reduzierte die Einrichtungszeit einer neuen Server-Instanz auf ein Minimum und wurde von Nutzern in der Folge sehr positiv bewertet. Der Mehrwert dieser Hilfsfunktion hat sich insbesondere im SVN gezeigt, da das dortige Deployment über die Jahre hinweg konsequent Updates erfahren hat und mittels dieses Assistenten ein kontinuerlicher Weiterbetrieb über Versionsgrenzen hinweg gewährleistet werden konnte.

Benutzerhandbuch : Ein wesentlicher Meilenstein war Ende 2016 die Veröffentlichung eines umfangreichen Benutzerhandbuchs für die gesamte HoneySens-Plattform. Es diente Administratoren als Einführung in den Umgang mit dem System und auch als Referenz bei auftretenden Fragen.

Meilensteine 2017

Modularisierung der Honeypot-Dienste : Die Entwicklungen in dieser Projektphase drehten sich überwiegend um zwei zentrale Fragen, eine davon war die Integration einer Vielzahl zusätzlicher Honeypot-Projekte in die Sensorsoftware. Der bisherige Ansatz, diese einfach als Bestandteil der Sensor-Firmware auszuliefern, erwies sich aufgrund des großen Platzbedarfes und der zusätzlichen

Update-Problematik als nicht weiter tragbar. Aus diesem Grunde wurde eine neue Architekfür für die Sensor-Software konzipiert, in der die Honeypot-Software in Form von sog. *Diensten* innerhalb von Containern betrieben wird. Diese sind leicht austausch- und aktualisierbar. Zusätzlich müssen auf den Sensoren nur noch die Dienste vorgehalten werden, die für den Betrieb tatsächlich erforderlich sind. Die anschließende Implementierung der Architektur zog weitrechende Modifikationen sowohl an der Server- als auch der Sensorsoftware nach sich, womit dieser Meilensteine und die damit verbundenen Herausforderungen die überwiegende Zeit des Jahres 2017 in Anspruch genommen haben. Das Ergebnis hat sich unterdessen bewährt: Honeypot-Dienste können inzwischen über die Webschnittstelle des Servers komfortabel hochgeladen, verwaltet und frei auf die Sensoren verteilt werden. Das somit nun verfügbare Repository an Diensten wächst v.a. durch die Mitarbeit der T-Systems MMS beständig weiter an.

Virtuelle Sensoren : Ein weiterer Meilenstein, für den die Grundpfeiler noch im Jahr 2017 gelegt wurden, war die Virtualisierung von Sensoren. Konkret wuchs der Bedarf an einer einheitlichen plattformunabhängigen Lösung, mit der Sensoren in virtuellen Maschinen platziert werden können. Der Bedarf wurde v.a. im Hosting-Bereich geäußert, wo komplett virtualisierte Netze einen physischen Port für die BeagleBone-Boards nicht mehr vorsehen. Die bisherige Lösung, dafür ein Debian-Paket bereitzustellen, hat sich als unzureichend erwiesen, da damit kein vollautomatisches Update möglich ist. Aus diesem Grunde wurde ebenfalls und in weiten Teilen auch parallel mit der Neugestaltung der Sensor-Architektur für die Honeypot-Dienste ein Konzept erarbeitet, wie Sensoren als virtuelle Plattform betrieben werden können. Das Ergebnis ist die Bündelung aller Sensoraktivitäten in einem schlicht als *Manager* bezeichneten Dienst, der alle plattformunabhängigen Funktionen der Sensoren abbildet. Zusätzlich kann dieser um Komponenten erweitert werden, die wiederum von Architektur und Plattform abhängig sind, beispielsweise spezifische Update-Skripte. Zusätzlich wurden die Sensorkomponenten während der späteren Projektabschnitte in mehreren Schritten so angepasst, dass ein Betrieb innerhalb eines Docker-Containers möglich ist, der schlussendlich auch bequem über Docker Compose verwaltet werden kann. Bis dahin war es ein langer Weg, an dessen Ende nun aber komfortabel nutzbare virtuelle Sensor-Container stehen, die ihre Flexibilität nicht nur im produktiven Einsatz, sondern insbesondere auch als Live-Entwicklungsplattform unter Beweis stellen.

Meilensteine 2018/2019

DevOps : In den letzten Projektabschnitten lag der Fokus vermehrt auf der Sicherstellung der künftigen Weiterentwicklung. So wurden sowohl für den Server als später auch die Sensoren Entwicklungsumgebungen geschaffen, die reproduzierbar aus einer bestehenden Codebasis gebaut werden können. Es genügt dafür der Aufruf eines simplen Befehls `make dev` im Server- oder Sensorverzeichnis, um eine entsprechende Testumgebung für die weitere Entwicklung zu erhalten. Nicht zuletzt auch der Wechsel vom klassischen privaten SVN hin zu einem Git-Repository, auf das alle Beteiligten Zugriff haben, haben die Zusammenarbeit im Team deutlich erleichtert.

Container-Orchestrierungssysteme : Betreiber komplexer containerbasierter Cloud-Infrastrukturen setzen vermehrt auf Orchestrierungsssteme wie Apache Mesos, Kubernetes oder Docker Swarm, um ihre Services effizient und flexibel zu betreiben. Auch innerhalb der zentralen Dienste des SVN kommt Docker Swarm zum Einsatz und gilt inzwischen als Referenzplattform für Dienstleister, die einen Service innerhalb des Verwaltungsnetzes anbieten wollen. Da auch der HoneySens-Server

als Container ausgeliefert wird, waren hier eine Reihe von Anpassungen nötig, damit dieser innerhalb einer Swarm-Umgebung betrieben werden kann. Das beinhaltet u.a. das vollständige Logging aller Dienste auf die Standardausgabe, eine separate Spezifikation der Diensteigenschaften und Heartbeat-Support.

Serverbetrieb hinter einem TLS-Proxy : Nachdem die Anpassungen für Docker Swarm vollständig integriert und getestet waren, konnten die Sensoren zunächst keine Verbindung zum Server aufbauen. Eine genauere Untersuchung ergab, dass vor dem Swarm-Cluster ein Proxy-Server alle eingehenden Verbindungen annimmt und an die entsprechenden Container weiterleitet, dabei jedoch auch TLS-Endpunkt ist und danach nur noch unverschlüsselt kommuniziert. Bei diesem Vorgehen wurden jedoch die TLS-Clientzertifikate, die Sensoren zur Authentifizierung am Server nutzen, schlicht verworfen. Recherche und eine Reihe von Tests führten letztendlich zu einer einvernehmlichen Lösung, bei der der TLS-Proxy Client-Zertifikate auswertet (und dafür Zugriff auf die HoneySens-CA benötigt) und das Ergebnis an die weitergeleiteten Anfragen anhängt.

PKI-Verwaltung : Die Certificate Authority, die jede HoneySens-Installation intern verwaltet, stellt standardmäßig Zertifikate mit der Laufzeit von einem Jahr aus. Nachdem einige der Test- und Produktivinstanzen über diesen Zeitraum hinaus betrieben wurden, stellte sich insbesondere nach der Implementierung zusätzlicher Sicherheitsfeatures auf Basis der Penetrationstests heraus, dass Zertifikate bereits ihr Ablaufdatum überschritten hatten und somit die Server-Sensor-Kommunikation korrekterweise nicht mehr möglich war. In einem recht aufwändigen Aktualisierungsprozess wurden zunächst händisch neue gültige Zertifikate erstellt und verteilt. Ein finaler Schritt beinhaltete schließlich die Konzeption und Implementierung einer rudimentären Verwaltung der CA: Alle Zertifikate im System können nun über das Webinterface aktualisiert werden, die Sensoren können den Wechsel hin zum neuen Zertifikat dabei transparent und selbstständig verarbeiten - eine manuelle Interaktion des Nutzers mit den Sensoren ist nicht nötig.

Honeypot-Repository : Seitdem Honeypot-Dienste mittels einer Containerumgebung modularisiert wurden, hat die kontinuierliche Pflege und Erweiterung des Repositories an verfügbaren Honeypots Priorität. Es wurden in der Folge weitere OpenSource-Honeypot-Projekte integriert und bestehende aktualisiert, darunter Honeypots für die Protokolle SSH, SMB, RDP, HTTP(S), aber auch für diverse Industrieprotokolle.

Veröffentlichung : Wie mit allen am Projekt beteiligten Parteien vereinbart, steht die letzte HoneySens-Revision der Öffentlichkeit als OpenSource-Projekt zur Verfügung. Hierfür wurde zunächst im Frühjahr 2018 der Versionsstand 0.2.5 als *Legacy*-Version auf GitHub veröffentlicht[5]. Ein Jahr später, im Januar 2019, erfolgte schließlich die Veröffentlichung der *HoneySens Community Edition*[6], die kontinuierlich gepflegt wird und mit einem von der MMS-internen Revision getrennten Versionsschema Neuerungen aus der proprietären Variante schrittweise übernimmt.

[5] `https://github.com/sylencecc/honeysens-legacy`
[6] `https://github.com/sylencecc/honeysens-ce`

4 Nachwort

HoneySens hat sich dank dieses vom Freistaat Sachsen finanzierten Forschungsprojektes von einer Idee, eine Plattform zur flächigen Verteilung von Honeypots in heterogenen Netzstrukturen zu schaffen, über eine prototypische Implementierung nach einer Diplomarbeit hin zu einem Produkt entwickelt, das gegenwärtig in zahlreichen IT-Landschaften eingesetzt wird. Auch außerhalb des kommerziellen Betriebs wird HoneySens in Unternehmen auf Basis der inzwischen frei verfügbaren Variante eingesetzt. Weiterhin strebt die TU Dresden für 2019 den Übergang von der seit 2016 andauernden Forschungsinstallation in den Produktivbetrieb an, womit auch genug Raum zur Integration weiterer innovativer Ideen aus der aktuellen Forschung im inzwischen wieder gefragten Feld der *Deception Networks* bleibt.

Im Zuge dieser Entwicklungen erklären wir das „Forschungsprojekt zur Entwicklung eines Sensornetzwerks, um Angriffe auf die IT-Infrastruktur des Freistaats zu erkennen“ - oder kurz schlicht *HoneySens* - als erfolgreich beendet und freuen uns, auf Basis der Resultate auch in Zukunft eine stabile Honeypot-Plattform zur Verfügung stellen zu können. In Namen der Professur für Datenschutz und Datensicherheit der TU Dresden danken wir allen Beteiligten für die Zusammenarbeit, insbesondere aber natürlich den Initiatoren aus dem Freistaat Sachsen, ohne deren Interesse, Geduld und natürlich finanzielle Unterstützung dieses Projekt nicht möglich gewesen wäre: dem Beauftragten für Informationssicherheit des Freistaats Sachsen, Herr Karl-Otto Feger und Herr Christoph Damm, der inzwischen das SAX.CERT leitet.

5 Anhang

Diese Auflistung soll die zuvor beschriebenen Prozesse und Veränderungen der letzten Projektphase anhand der im Laufe des Forschungsprojektes in der Versionsverwaltung hinterlegten Kommentare für jede neue Softwarerevision dokumentieren. Die Revisionsnummern werden in dieser Variante nur noch exemplarisch weitergeführt, da inzwischen der Umzug vom zuvor genutzten SVN hin zu Git erfolgt ist, das bewährte Format dieses Anhangs aber beibehalten werden soll.

Revision	**Änderungen**
582	Fixed a bug that caused event mails to be sent to recipients regardless of their division assignment
583	Fix: Incident messages now respect event filters
584	UI: Service checkboxes are now disabled by default and can be enabled with a separate button to reduce the risk of accidentally modifying service assignments
585	UI: Service checkbox toggle label adjusted; bug fixed that caused checkboxes to be re-enabled during background state updates
586	gitignore updates to match current repository structure
587	Sensor launcher restart policy added
588	Dev: grunt-chokidar added as watch backend and set as default, alternatives can be enabled within the Makefile
589	Server distribution packaging now contains installation instructions and a compose file next to the server docker image
590	Fixed a bug that caused mails to be sent out to contacts that weren't supposed to receive them
591	- Server package archive now extracts a directory instead of a bunch of files - Installation documentation fixed
592	Dockerized sensors deployment is now based on docker compose, including a self-updating mechanism
593	Dockerized sensors: - Support for a 'development sensor' instance that restarts when the manager code is modified - CONFIG_FILE parameter removed, we now treat the first file in the configuration directory as input config

594	Dockerized sensor: - Honeypot network interface now configurable - Sharing the network stack with the host is now the default setting - Host network stack configuration (netfilter etc.) is now performed by the manager - Clean manager shutdown process implemented
595	Sensor manager: - Service containers now populate a separate 'services' network - Shutdown can now be initiated via the CLI (and still SIGTERM) - „skip init" switch name changed to „dev mode" Dockerized sensor: - Performs a cleanup of the services network after container shutdown - Workaround added that properly shuts down dockerd
596	Support for custom service network ranges added (default: 10.10.10.0/24)
597	Dockerized sensor revision bumped to 1.1.0: - Build files updated so that the resulting firmware archive contains all resources necessary for deployment - Documentation updated - Launcher removed - .gitignore updates
598	Fix: Sensor removal with active service assignments
599	Fix: Server skipped some events due to overlong data fields, those are now truncated automatically
600	Fix: Dockerized sensor firmware is not packaged with a top-level directory anymore, because that caused issues with automatic processing of the archive by both server and sensors
601	Sensor manager: Colorful, more verbose log output (including timestamps)
602	Sensors now provide feedback about their service's status and display that in the web UI
603	Sensor manager: - State management outsourced into a separate thread, ensuring consistent sensor feedback throughout service or firmware updates - Log level can now be specified via the compose file (if omitted, it defaults to 'info')
604	Sensor manager event caching
605	UI: Event list columns ID, summary and status made sortable
606	BeagleBone auditing with USB mass storage devices support (insert a writeable USB stick to receive log data)

607	UI: The event list now summarizes the amount of „details“ per event
608	- LED support for the BBB platform reworked: now controlled by the manager and supports various notification modes (server reachability, event status, updates and audit) - Sensor manager CLI can now be instructed to change the logging level and control the LED on BBBs Fix: Removed reference to led.sh in build scripts LED notification on USB audit Different LED notifications for server unreachability, firmware or service updates - More responsive LED notifications on service downloads - Streamlined LED logic into a single function
609	UI: Form validation rework, including a bunch of usability improvements
610	UI: Basic event list filter added
611	Fix: Mail configuration form and weekly distribution
612	Fix: Server update script now properly adds the sensor network to the configuration file
613	Fix: recon dependencies
614	Fix: Server update script now properly triggers the update step from 1.0.4 to 2.0.0
615	Service cowrie: Build process optimizations
616	UI: Upload forms homogenized
617	UI: Event search field resized, placeholder text stated more precisely
618	UI: Open external links in a new tab or window
619	Fix: Template literal workaround for IE
620	Fix: Form feedback no overlays help popovers
621	Fix: Visibility of service labels in the sensor list
622	Fix: UI now allows the creation of up to 1024 sensors and filters
623	Fix: Made sensor manager more resilient when talking to an outdated server
624	UI: Show uploaded file name after a service was registered successfully