Technische Universität Dresden

Fakultät Informatik

Professur für Datenschutz und Datensicherheit

Projekt HoneySens

9. Zwischenbericht

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

Stand: 30.11.2017

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.

Die gegenwärtige Projektphase hatte primär zum Ziel, die beiden im vorangegangenen Bericht angesprochenen Thematiken der Modularisierung von Sensor-Diensten im Rahmen des Forschungsprojektes und die Virtualisierung von Sensoren im Rahmen der Weiterentwicklung des Projektes durch die T-Systems Multimedia Solutions (MMS) zusammenzuführen und bis Jahresende in einer gemeinsamen Version *MMS 1.0* bereitzustellen. Mit Erreichen dieses Meilensteins ist weiterhin auch die parallele Veröffentlichung der letzten Rollout-Version 0.2.4 als OpenSource-Version geplant, in der insbesondere die Virtualisierung als exklusives Feature für MMS-Kunden noch nicht enthalten sein wird. Ergebnisse aus dem Forschungsprojekt fließen, soweit sinnvoll, vollständig in die OpenSource-Version ein. Für die weitere Entwicklung nach dem genannten Meilenstein wird ein öffentliches Repository für die OpenSource-Version und ein darauf basierendes MMS-internes Repository mit zukünftigen Entwicklungen, die zu einem späteren Zeitpunkt in die offene Version einfließen werden, Verwendung finden. Ein Zweig dessen wird zudem ausschließlich optische Anpassungen für die Kunden der MMS-Version beinhalten und nicht gesondert veröffentlicht werden.

Um die Zusammenführung und den Fortschritt hin zum Erreichen der Version MMS 1.0 nachvollziehbar beschreiben zu können, wird dieser Forschungsbericht Entwicklungen sowohl aus dem Forschungsprojekt als auch innerhalb der MMS aufgreifen und näher erläutern. Dies betrifft insbesondere die Virtualisierung der Sensoren, der in diesem Bericht ein gesonderter Abschnitt eingeräumt wird. Grund hierfür ist, dass hinter Modularisierung und Virtualisierung in weiten Teilen zusammenhängende Änderungsprozesse stehen, die für sich nicht isoliert betrachtet werden können. Beide Änderungen resultieren in einer vollständig überarbeiteten Software für die Sensoren, die in Zukunft leicht auf neue Plattformen portiert werden kann und zudem viele Schwierigkeiten bei der Interaktionen zwischen den lokalen Sensor-Komponenten mit Hilfe des verwaltenden Manager-Daemons elegant löst.

Ein weiterer Abschnitt wird die Änderungen in der kommerziellen Version zusammenfassen und weitere, in dieser Projektphase erzielte Fortschritte im Forschungsprojekt aufzählen, die alle Gegenstand der Meilenstein-Version MMS 1.0 sein werden. Noch nicht integrierte Entwicklungen aus dem Forschungsprojekt werden an dieser Stelle ebenfalls genannt.

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 2017 bis zum 30. November 2017 verzeichneten Fortschritte im HoneySens-Projekt. Teilbereiche, die in den Anforderungen des ersten Projektberichtes genannt, aber hier nicht weiter beschrieben werden, sind Gegenstand der zukünftigen Weiterentwicklung.

2.1 Modularisierung von Honeypot-Diensten

Das grundsätzliche Konzept für die Modularisierung, nämlich die Auslagerung der zumeist quelloffenen Honeypot-Dienste in Docker-Container innerhalb der Sensoren mit dem Ziel, die Sensoren besser an die umgebende IT-Landschaft anzupassen und einzelne Honeypot-Software gezielt hinzuzufügen, aktualisieren oder entfernen zu können, wurde bereits im letzten Projektbericht ausführlich vorgestellt. Die damalige Umsetzung beschränkte sich primär auf alle erforderlichen Änderungen auf Serverseite beinhaltete nur eine prototypische Implementierung der dazu passenden Sensor-Software. Dieser Punkt wurde in dieser Phase aufgegriffen: Der Manager-Daemon, der die Basisfunktionalität eines jeden Sensors stellt - darunter der regelmäßige Kontakt zum Server, Netzwerk-Konfiguration, Logging und Ereignisverarbeitung - wurde nun um eine *Services*-Komponente erweitert. Diese initialisiert einen lokalen Docker-Daemon und steuert das Zusammenspiel zwischen Sensor und der internen Docker-Registry, die auf der Seite des Servers vorgehalten wird. Konkret bedeutet das, dass dem Sensor bei jedem Polling-Vorgang vom Server eine Liste der zu betreibenden Honeypot-Dienste respektive Revisionen zur Verfügung gestellt wird, auf deren Basis anschließend ein Vergleich mit den lokal vorhandenen Services erfolgt. Neue Services, bzw. neue Revisionen werden bei Bedarf vom Server bezogen, intern weiterzuleitende TCP- oder UDP-Ports für die Dienste automatisch aus dem Docker-Template extrahiert und anschließend der Startvorgang ausgelöst. Auch das Aktualisieren oder Beenden bereits laufender Dienste gehört in den Verantwortungsbereich der Komponente.

Da Sensoren auch zukünftig abhängig von ihrer Platzierung an das umgebende Netzwerk angepasst werden sollen, um bei Angreifern keinen Verdacht der beabsichtigten Täuschung zu wecken, ist die individuelle Konfiguration der aktiven Dienste pro Sensor wünschenswert. Das war mit der bisherigen Architektur von statischen Sensordiensten bereits möglich, musste nun aber an die dynamische Service-Architektur angepasst werden. Sogenannte *ServiceAssignments* weisen nun jedem Sensor die Dienste und zugehörige Revisionen zu, die betrieben werden sollen. Deren Konfiguration kann zudem im überarbeiteten Web-Interface leicht angepasst werden. Es wurde außerdem beim Entwurf im Backend bereits berücksichtigt, dass in Zukunft auch jeder Dienst eigene Konfigurationsoptionen bereitstellen kann (um beispielsweise einzelne Features eines 3rd-Party-Honeypots zu aktivieren/deaktivieren). Die Implementierung dieses Features war allerdings noch nicht Gegenstand dieser Projektphase, wird aber bei einer zunehmenden Größe des Repositories an verfügbaren Honeypot-Services schnell an Bedeutung gewinnen.

Die Erreichbarkeit der serverseitigen Low-Interaction-Honeypots muss weiterhin von der Sensor-Firmware gewährleistet werden. An diesem Punkt kommt die Virtualisierung, bzw. Modularisierung der Sensor-Plattform ins Spiel, die im Rahmen der MMS entwickelt wurde. Deren Umsetzung wird im nachfolgenden Kapitel detaillierter beschrieben.

2.2 Die Sensor-Plattform: Modularisierung und Virtualisierung

HoneySens wurde ursprünglich auf Einplatinencomputer des Typs BeagleBone Black zugeschnitten. Diese boten zum Zeitpunkt der Konzeptionierung des Systems eine Reihe vorteilhafter Eigenschaften (nebst der guten Verfügbarkeit), die für ein Sensorsystem genutzt werden konnten. Im Rahmen des praktischen Einsatzes innerhalb produktiver Netze hat sich jedoch herausgestellt, dass derartige physische Deployments bei einer zunehmenden Virtualisierung von Netzen in Rechenzentren zunehmend schwierig, bzw. ganz unmöglich werden. Auch im Rahmen der sich verstärkt ausbreitenden Techniken wie *Software Defined Networking* (SDN) und *Network Function Virtualization* (NFV) ist abzusehen, dass physische Ports für Sensoren in Zukunft nicht mehr zur Verfügung stehen werden. Die MMS hat zudem Kundenfeedback auf Messen und Infoveranstaltungen gesammelt, das sich mit diesen Beobachtungen deckt. Aus diesen Gründen wurde beschlossen, den Fokus der Weiterentwicklung bei der MMS zunächst auf flexiblere Sensor-Deployments zu lenken. Konkret bedeutet das, im gesamten HoneySens-Projekt alle BeagleBone-spezifischen Abhängigkeiten in austauschbare Komponenten auszulagern, so dass in Zukunft auch spezielle Funktionalität für andere Plattformen hinzugefügt werden kann. Zusammengefasst findet also auch hier eine Modularisierung statt. Die erste zusätzliche Plattform neben dem BeagleBone Black (das in dieser Form auch weiterhin unterstützt wird) wird eine allgemeine virtuelle Sensorplattform sein, die bspw. in virtuellen Maschinen zum Einsatz kommen kann.

Serverseitige Modularisierung Das HoneySens-Backend kannte bisher das Konzept der Firmware in Form von vom Administrator bereitgestellten Images für SD-Karten, die ausschließlich zur Installation neuer BeagleBone-Sensoren genutzt werden konnten. Um diesen Ansatz zu generalisieren, wurde das Konzept der Plattform (engl. *Platform*) eingeführt, die eine Hardware- oder Softwareplattform beschreibt, auf der Sensoren betrieben werden können. Die von HoneySens unterstützen Plattformen sind abhängig von der Version des HoneySens-Servers und können vorerst nicht dynamisch ergänzt werden. Wenn sich hierfür in der Praxis ein interessanter Use-Case ergeben sollte, wäre der Schritt hin zu dynamischen, bzw. individuellen Plattformen allerdings trivial. Weiterhin ist mit jeder Plattform eine Liste an Firmware-Revisionen verknüpft, aus denen Nutzer dann wie gewohnt eine Version zum Standard für alle Sensoren erklären können.

Abgesehen von den nötigen zusätzlichen Dialogen und Modulen für die webbasierte Benutzerschnittstelle waren die serverseitigen Änderungen zur Unterstützung modularer Plattformen allerdings überschaubar. Über einen neuen API-Endpunkt `/api/platforms` können Plattformen und Firmware eingesehen und verwaltet werden, die Implementierung orientiert sich dabei stark an der bestehenden Firmware-Verwaltung. Da auf allen Plattformen identische Sensorsoftware zum Einsatz kommt (bzw. diese aus der gleichen Codebasis erzeugt wird), ist zudem der Prozess zum Hinzufügen und Editieren von Sensoren unabhängig von der zum Einsatz kommenden Sensorplattform. Nachdem ein Administrator einen neuen Sensor registriert hat, erhält er vom Server wie gewohnt ein Archiv, das erst im nächsten Schritt für die Einrichtung des Sensors genutzt werden muss. Es ist dabei irrelevant, welche Plattform zum Einsatz kommen soll: Das Format des Konfigurationsarchives ist an dieser Stelle standardisiert. Ein Wechsel zwischen Plattformen, um beispielsweise einen physikalischen Sensor in einen virtuellen umzuwandeln, ist auf diesem Weg auch problemlos möglich.

Virtuelle Sensoren Die erste neue Instanz des Plattform-Konzeptes soll es ermöglichen, Sensoren virtualisiert und somit unabhängig von der zugrundeliegenden Hardware zu betreiben. Hierfür wurden verschiedene Varianten der Virtualisierung in Betracht gezogen, schlussendlich aber die Entscheidung zugunsten einer containerbasierten Lösung gefällt. Ausschlaggebend war, dass es im professionellen Umfeld keine Standard-Virtualisierungslösung gibt, sondern Produkte von VMware, Microsoft oder Oracle alle gleichermaßen verbreitet sind. Virtualisierte Sensoren auf nur eine oder wenige solcher Plattformen zuzuschneiden würde den Einsatz in einigen IT-Infrastrukturen erheblich erschweren und zudem einen hohen Pflegeaufwand nach sich ziehen. Der Betrieb eines Linux-Systems innerhalb von Containern mit Docker, LXC usw. gestattet Nutzern hingegen größte Flexibilität bei der Auswahl der zugrundeliegenden Virtualisierungsplattform und ermöglicht zusätzlich auch den Bare-Metal-Betrieb von Sensoren

Das Konzept sieht vor, dass die gesamte Sensorsoftware innerhalb eines Docker-Containers betrieben wird. Einzige Anforderung beim Betrieb eines solchen Sensors ist folglich die Unterstützung von Docker-Containern durch das Betriebssystem. Hier kann jede aktuelle Linux-Distribution zum Einsatz kommen, da der nötige cgroups-Support typischerweise zur Standardkonfiguration des Kernels gehört. Da die Honeypot-Software selbst auch wieder gekapselt auf Docker-Basis betrieben wird, müssen außerdem Container innerhalb von Containern betrieben werden, was - die entsprechenden Privilegien vorausgesetzt - kein Hindernis ist.

Innerhalb des Sensor-Containers kommt wie auch auf den BeagleBones der Manager-Daemon zum Einsatz, der im Rahmen der Modularisierung der Honeypot-Dienste entstanden ist und nun die zentrale Management-Instanz der Sensoren darstellt. Die Aufgaben dieses Prozess wurden bereits im 8. Zwischenbericht ausführlich vorgestellt. Jeglicher plattformspezifischer Code - wie beispielsweise die LED-Ansteuerung beim BeagleBone Black - wurde in gesonderte Plattform-Komponenten ausgelagert.

Um unterschiedliche Prozesse gleichzeitig innerhalb eines Docker-Containers verwalten zu können, bedarf es zudem eines rudimentären Init-Systems, das beispielsweise verwaisten Zombie-Prozessen vorbeugen kann. Die Wahl fiel hier auf *s6*, eine minimale Lösung, die laufende Dienste statt mit PID-Files mit Supervisor-Prozessen überwacht, alle nötigen Primitive zur Prozessverwaltung integriert und zudem bereits in Basis-Templates auf dem Docker Hub zur Verfügung steht[1]. Neben Start- und Stop-Skripten für alle auf dem Sensor benötigten Dienste stellte lediglich die Behandlung der Datei `/etc/resolv.conf` eine Herausforderung dar: Diese wird vom s6 Init-System direkt verwaltet, für die erfolgreiche Kommunikation mit dem HoneySens-Server sind jedoch bei der Verwendung von selbstsignierten Zertifikaten Modifikationen an dieser Datei erforderlich. Ein Workaround wurde mit Hilfe eines zusätzlichen lokalen DNS-Daemons `dnsmasq` geschaffen, der als eine Art Proxy fungiert und in die Namensauflösung korrigierend eingreift. Weiterhin musste sichergestellt werden, dass alle Komponenten auf dem Sensor, darunter auch der Service-Docker-Daemon, korrekt Proxy-Server verwenden, falls diese vom Administrator für einen Sensor konfiguriert wurden.

Damit die Erreichbarkeit des Sensors innerhalb eines Docker-Containers sichergestellt werden kann, wurde die Entscheidung zugunsten eines Dual-Interface-Ansatzes gefällt: Der Container des Sensors erhält zusätzlich zum Standardnetzwerk einen weiteren virtuellen Netzwerk-Adapter, der über eine Software-Bridge auf dem Hostsystem mit dem externen Interface des darunterliegenden Hostsystems verbunden ist. Somit kann der Sensor im Container direkt („raw“) mit dem physischen Netzwerkadapter des Hosts arbeiten und alle eingehenden und ausgehenden Daten verarbeiten, ohne dass diese vom IP-Stack des Hostsystems zuvor gefiltert werden. Mit diesem Ansatz ist sichergestellt, dass die Netzwerkkonfigurati-

[1] `http://www.skarnet.org/software/s6/index.html`

on über das Webinterface unabhängig von der Konfiguration des zugrundeliegenden Hostsystems vorgenommen werden und dass der recon-Dienst, der auf einen „Raw“-Socket angewiesen ist, weiterhin zum Einsatz kommen kann. Zusätzlich bereitet diese Maßnahme auch den zukünftigen Einsatz von `honeyd` vor, einem Honeypot-Framework, das im Zusammenhang mit der Erweiterung der Sichtbarkeit von Sensoren in Zukunft Verwendung finden wird.

Für den Betrieb des Docker-Sensors sind zudem eine Reihe von Management-Skripten erforderlich, die auf dem Hostsystem laufen und beispielsweise ein Update des gesamten Sensor-Containers vornehmen, wenn dies vom Server angefordert wird.

2.3 Zusammenführung Forschungsergebnisse und MMS-Entwicklung

Für den beabsichtigten Übergang in den Dauerbetrieb wurden und werden weiterhin bis Jahresende alle sowohl im Rahmen des Forschungsprojektes als auch durch die T-Systems MMS vorgenommenen Entwicklungen am Projekt zusammengeführt und in der Version *MMS 1.0* gebündelt. In diese fließen alle Fortschritte ein, die im Rahmen der vergangenen Zwischenberichte bereits genauer ausgeführt wurden. Nicht Teil dieses Meilensteins ist lediglich die Erweiterung der Sichtbarkeit von Sensoren durch intelligentes ARP-Spoofing, da sich diese an einer früheren Stelle bereits beschriebe Idee noch immer in der Planungsphase befindet und noch nicht prototypisch umgesetzt wurde. MMS-Entwicklungen umfassen insbesondere große Teile des virtualisierten Sensors sowie die Möglichkeit für den Benutzer, mehrere Ereignisse auf einmal zu Bearbeiten oder zu Löschen, mehr Flexibilität und Bugfixes für E-Mail-Notifikationen und Usability-Verbesserungen wie beispielsweise die dynamische Anpassung der Ereignisliste an die Displaygröße. In diesem Rahmen ist zudem für eine Demonstration ein prototypisches Conpot-Modul[2] entstanden, das zukünftig als Grundlage für einen Conpot-Integration als Service dienen wird. Um den Wechsel auf die virtualisierte Sensorplattform für Kunden zu beschleunigen, wurde zudem ein Demonstrator entwickelt, mit dem ein Sensor nativ (ohne die oben beschriebene Container-Isolation) auf einem Debian-System als Teil der Paketverwaltung installiert und betrieben werden kann.

[2] `https://github.com/mushorg/conpot`

3 Ausblick

Bis Jahresende ist, wie bereits angesprochen, eine Veröffentlichung der Version *MMS 1.0* geplant. Für die in diesem Bericht genannten Punkte, insbesondere die Virtualisierung, sind noch weitere Testreihen und Versuche geplant. Zu einem späteren Zeitpunkt ist zudem ein professioneller Penetrationstest beabsichtigt, um den aktuellen Stand für den praktischen Einsatz vorzubereiten.

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

Erweiterung der Honeypot-Funktionalität : Mit Fertigstellung der modularen Honeypot-Services existiert nun ein standardisiertes Containerformat, in dem leicht neue OpenSource-Honeypot-Projekte in HoneySens eingebunden werden können. Hier gilt es, eine sinnvolle Vorauswahl zu treffen und in produktiven Netzen häufig anzutreffende Dienste abzubilden.

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.

Ereignisauswertung : Aus dem Praxisbetrieb insbesondere an der TU Dresden stammt die Erkenntnis, dass die Auswertung der gesammelten Ereignisdaten mit zunehmender Menge Schwierigkeiten bereitet. Die Ereignisliste reicht dann als alleiniges Mittel zur Verarbeitung der Daten nicht mehr aus und es gilt, darüber nachzudenken, wie die Daten sinnvoller aggregiert und dem Nutzer visualisiert werden können, so dass nur noch für Detailfragen auf die Ereignisliste zurückgegriffen werden muss. Weiterhin ist auch eine Anbindung an SIEM-Systeme in Betracht zu ziehen, die Honeypot-Daten mit Informationen aus anderen Quellen korrellieren 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
433	- Checkboxes added to event list for mass edit/removal (not yet functional) - Missing web fonts added
434	- Event list now scales with available vertical space
435	- Options panel added for multiple selected events
436	- Event paginator repositioned (aligned to the right) - Event options panel is now properly shown when events are selected - backbone.paginator updated for additional event support - Fixed a few bugs that may cause problems when events are manually triggered on collections
437	- Mass-deletion of events backend implementation and frontend connection established - Mass-edit UI for events added
438	- Highlighting of selected events within the event list
439	- Documentation for the events/delete method
440	- Backend for mass event updates (status code) implemented and connected to frontend
441	- Registry status indicator added to web frontend
442	- Server snakeoil certificate generation routine changed so that CA certs are generated - Documentation update
443	Sensor manager: - Services manager will now register the registry certificate
444	- Docker TLS key generation automation
445	- Sensor manager compatibility fixes with the private HoneySens registry
447	- Sensor API resource parameter naming is now more consistent - ServiceAssignments added as new entity to reference active services from sensors - Sensor list updated to include service assignment - Sensor manager preparations for dynamic service handling
448	- Sensor manager now properly handles the creation and deletion of service containers - Fixed a bug within the self-signed cert creation script
449	- A default revision can now be set per service (frontend + backend)

450	- Baremetal added as a temporary platform to run the 0.2.3 firmware on a native debian host, but without the dionaea service.
451	- Removing a service now also removes all its revisions
452	Server version 0.2.4: - „Send ALL events“ functionality added for contacts - Successful test mails now yield proper positive feedback - Service functionality of previous commits included, buy disabled (not yet ready for production)
453	- Preliminary conpot module
454	- Fixed a logging bug in the kippo module
455	- Services features re-enabled in the frontend - Docker daemon added to server container - Fixed a bug that prevented services from being properly removed
456	- Generic sensor base image updated (implying an update to py-docker) - Manager service handling improved to fit the updated py-docker - Cowrie metadata added - Sensor config creation worker updated to work with relative paths
457	Dockerized sensor container: - Proper updates of /etc/hosts and /etc/resolv.conf - S6 init system added with manager and docker as services - Proxy support for the nested docker instance
458	Sensor manager: - System time update moved to the bbb platform - New hook added: ON_BEFORE_POLL - Docker platform: Manager output is now unbuffered - Smaller adjustments for code and output readability - External launch scripts added to manage the dockerized sensor
459	- Sensor manager: dummy platform module added
460	Sensor manager: - Docker platform now reconfigures its honeypot network interface - dhcpd added to virtual sensors - Debinterface sources replaced by a PyPi dependency
461	Backend: - ServiceManager introduced as proxy for the service layer - Platform services skeleton added