Technische Universität Dresden

Fakultät Informatik

Professur für Datenschutz und Datensicherheit

Projekt HoneySens

1. Zwischenbericht

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

Stand: 16.07.2015

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. Hauptaugenmerk des Forschungsprojektes ist weiterhin die Koordinierung der vielfältigen Anforderungen der am Projekt interessierten und teilnehmenden Ressorts der Landesverwaltung.

Die Analyse und Konzeption dieser Anforderungen erfolgte bereits im Vorfeld des Forschungsprojektes im Rahmen einer Diplomarbeit zum Thema "Realisierung eines Honeypot-Netzes", die ebenso eine prototypische Implementierung der genannten Konzepte beinhaltete. Als Architektur wurde ein verteiltes Sensornwetzwerk gewählt, das alle auftretenden Ereignisse auf einer zentralen Serverinstanz speichert, die zudem eine graphische Benutzeroberfläche zur komfortablen Wartung und Verwaltung anbietet. Diese *"HoneySens"* getaufte Umsetzung wurde zudem in verschiedenen produktiven Netzwerkumgebungen getestet und hat sich als eine probate, erweiternswerte Lösung hervorgetan. Die erfolgreiche Präsentation des Projektes im Rahmen des SID Kundenforums 2014 und auf dem Gemeinschaftsland des IT-Sicherheitsrates auf der CeBIT 2015 waren ebenfalls dem Interesse zuträglich, das Projekt weiterzuverfolgen und noch offene Fragen im Rahmen eines Forschungsprojektes zu beantworten. Dieser Zwischenbericht soll über die bis zum 16. Juli 2015 durchgeführten Änderungen und Erweiterungen der Architektur informieren.

2 Anforderungen

Bei den gegen Ende der Diplomarbeit, sowohl im Labor als auch in praktischen Umgebungen, durchgeführten Testreihen stellten sich zunächst eine Reihe von Schwächen und noch offenen Problemen mit der prototypischen Implementierung heraus. Hierzu zählten insbesondere eine fehlende Möglichkeit zur Filterung aufgezeichneter Ereignisse. Je nach Netzwerkumgebung war nicht auszuschließen, dass die Sensoren periodisch wiederkehrende Ereignisse aufzeichneten, die unter den gegebenen Umständen jedoch keine Gefahr darstellten und sich auch nicht ohne Weiteres verhindern ließen. Hierzu zählen beispielsweise aktive Systeme, die aus Sicherheitsgründen in regelmäßigen Abständen ein Gesamtbild des Netzwerkes zeichnen und hierzu mit allen Hosts Kontakt aufnehmen müssen. Ebenso können falsch konfigurierte Geräte oder diverse Anwendungssoftware dazu führen, dass regelmäßig größere Mengen von harmlosen Ereignissen aufgezeichnet werden, die schlussendlich dazu führen, dass die Übersicht in der webbasierten Benutzeroberfläche des Sensornetzwerks verloren geht und wirklich wichtige Vorfälle in der Masse der redundanten uninteressanten Ereignisse verschwinden. Die Integration eines Filtersystems, das ein **Whitelisting** von regelmäßigen, harmlosen Ereignissen ermöglicht, war folglich eines der unmittelbaren Ziele des Projektes.

Ein weiteres, eher langfristig relevantes Problem war die Spezialisierung des Prototypen auf eine fixe

Hardwareplattform für die Sensorgeräte. Die verwendeten Ein-Platinen-Computer namens *BeagleBone Black* der Firma *Texas Instruments* würden in Zukunft veraltet und möglicherweise als Nachfolger produzierte Geräte mit den Vorgängern nicht mehr kompatibel sein. Weiterhin war die Beschaffung der Geräte in größeren Mengen zu Projektbeginn fragwürdig und folglich eine Portierung der Sensorsoftware auf andere Plattformen wünschenswert. Mehr **Flexibilität** in der Auswahl der unterstützen Hardware würde den Aufbau einer großen HoneySens-Plattform innerhalb des Sächsischen Verwaltungsnetzwerkes deutlich erleichtern.

Im Rahmen des Forschungsprojektes war weiterhin vorgesehen, eine Liste von am Projekt interessierten Ressorts zu erstellen, die später für eine erste größere Testinstallation des Systems in Frage kommen würden. Dieser Schritt wäre zunächst notwendig, da die ans SVN angebundenen Ressorts administrative Hoheit über ihre jeweilige IT-Infrastruktur besitzen und einer Installation der HoneySens-Plattform zunächst zustimmen müssen. Aus diesem Wunsch erwuchs zugleich die Anforderung, das Gesamtsystem vollständig **mandantenfähig** zu gestalten, so dass mehrere voneinander unabhängige Sensorinstallationen in verschiedenen Teilnetzen nebeneinander, aber aus administrativen Gründen auf der gleichen Serverinstanz betrieben werden können. Die Installation, Wartung und Analyse des im Anschluss einzurichtenden ressortübergreifenden **Testsystems** innerhalb des Verwaltungsnetzwerks gehörte ebenfalls zu den mit den am Projekt Beteiligten abgestimmten Anforderungen. Da die Mandantenfähigkeit eine für deren ordnungsgemäßen Betrieb notwendige Voraussetzung ist, wurde dieser die höchste Priorität aller unmittelbaren Anforderungen zugewiesen.

Die prototypische Sensorsoftware enthielt neben einigen OpenSource-Honeypot-Diensten für weit verbreitete Netzwerkprotokolle wie beispielsweise *SSH* oder *SMB/CIFS* auch die Möglichkeit, alle Verbindungsversuche zu eigentlichen geschlossenen TCP-Ports und jegliche eintreffende UDP-Pakete aufzuzeichnen. Den auf diese Weise gesammelten Datensätzen war jedoch gemein, dass insgesamt nur wenig Informationen über einen potentiellen Angriff gewonnen werden konnten: Die Quelle, das verwendete Protokoll und der Zielport des Netzwerkpaketes. Zur leichteren Klassifikation eines solchen Vorfalls wäre es jedoch wünschenswert, zusätzlich auch den Inhalt - oder zumindest einen Teil davon - aufzuzeichnen. Die Konzeption und Implementierung eines Dienstes, der lokal auf jedem Sensor **TCP-Verbindungen annehmen** und die vom Gegenüber nach dem Verbindungsaufbau gesendeten Daten zum HoneySens-Server senden kann, wurde folglich ebenso in die Zielsetzung des Forschungsprojektes aufgenommen. Hierbei ist insbesondere die Resistenz gegen potentielle DoS-Angriffe auf diesen Dienst zu berücksichtigen.

Eine interessante Frage ist weiterhin die Konzeption eines Prozesses, der die auf den Sensoren laufende Firmware für den Administrator des Systems völlig transparent aktualisieren kann. Dies ist nötig, da die Grundlage der Sensorsoftware eine Variante der frei verfügbaren Linux-Distribution *Debian GNU/Linux* ist, die von einer weltweit agierenden Community regelmäßig mit aktuellen Versionen der mitgelieferten Software versorgt wird. Da diese Updates in vielen Fällen auch bestehende Sicherheitslücken schließen, ist es unabdingbar, bei Bedarf **neue Firmware** für die Sensoren erstellen zu können.

Es wurden schlussendlich noch eine ganze Reihe von Anforderungen mit untergeordneter Priorität beschlossen, die insbesondere die graphische Benutzeroberfläche betrafen. Hierzu zählen unter Anderem das Anbieten von allgemeinen Funktionen zur Wartung des Systems, beispielsweise um die Datenbank zurückzusetzen oder alle bereits gesammelten Ereignisse zu entfernen. Zusätzliche Steuerelemente zur leichteren Filterung der Ereignisliste sind wichtig, wenn diese nach längerem Betrieb eine große Menge von Vorfällen beinhaltet und darin effektiv gesucht und sortiert werden soll. Unter Ausnutzung von *La-*

zy Loading wäre es zudem möglich, auch mit sehr umfangreichen Ereignislisten noch schnell arbeiten zu können, da in einem solchen Fall nur die aktuell darzustellenden Ereignisse vom Server angefordert werden.

3 Sachstand

Dieser Abschnitt gibt einen Überblick über die bis zum 16. Juli 2015 durchgeführten Arbeiten am HoneySens-Projekt unter Berücksichtigung der zuvor genannten Anforderungen. Teilbereiche, die in den Anforderungen genannt, aber hier nicht weiter beschrieben wurden, sind Gegenstand der zukünftigen Weiterentwicklung.

Refactoring Der während der Diplomarbeit entstandene Code wurde zunächst den qualitativen Anforderungen eines professionellen Softwareprojektes nicht gerecht, da es sich hierbei lediglich um eine via *Rapid Prototyping* entwickelte Anwendung zur Verdeutlichtung der entwickelten Konzepte handelte. Um die zukünftigen, umfassenden Erweiterungen realisieren zu können, ohne auf die bisherigen Softwarebasis völlig zu verzichten, wurde zunächst ein umfassendes Refactoring der Projektstruktur und der wichtigsten Anwendungsbereiche begonnen. Hierbei handelt es sich um einen langanhaltenden Prozess, der noch immer nicht vollständig abgeschlossen ist, in den betreffenden Bereichen allerdings die Code-Qualität bereits deutlich steigern konnte. Dies äußert sich insbesondere in der Lesbarkeit des Programmcodes, der zukünftigen leichten Erweiterbarkeit durch neue Funktionalität und einer gesteigerten Performance, da beispielsweise die Webanwendung zur Steuerung der Software ihren gesamten JavaScript-Code in komprimierter Form mittels einer einzigen Anfrage vom Server beziehen kann, obwohl der Code während der Entwicklung der Übersichtlichkeit wegen auf unzählige einzelne Dateien aufgeteilt ist.

Der Refactoring-Prozess bestand hierbei abstrahiert aus zwei wesentlichen Schritten:

1. **Modularisierung** aller Programmteile, so dass einzelne Module ersetzt oder hinzugefügt werden können, ohne bestehende Funktionalität in anderen Modulen zu beeinflussen. Hierfür ist ebenfalls die Definition eindeutiger Schnittstellen zur Kommunikation zwischen Modulen von Bedeutung. Zunächst wurde dieser Schritt auf die diversen Teilbereiche der Webanwendung und der serverseitigen API angewendet, in Zukunft ist ein analoger Prozesse für die Sensor-Firmware geplant.
2. **Integration in das Build-System** Zur Vereinfachung des späteren Deployments war es unabdingbar, die Erstellung der finalen Anwendung aus dem Quellcode so weit wie möglich zu automatisieren. Hierfür wurden die Build-Systeme *Grunt* und *GNU make* installiert und entsprechende Skripte geschrieben, die diesen Prozess und alle zugehörigen Teilschritte beschreiben. Erstere Software kommt dabei für das Deployment der Web-API und -Anwendung zum Einsatz, während letztere für das Generieren von Sensorsoftware genutzt wird. Weiterhin ist es wichtig, die aus diesen Prozessen entstandene Software automatisch in einen *Docker*-Container verpacken zu können, der schlussendlich auf Produktivsystemen genutzt werden kann. Hierfür sind ebenfalls automatisierende Skripte und ein sogenanntes *Dockerfile* nötig. Die aus dem vorherigen Schritt gewonnenen Module wurden alle ebenfalls in den Build-Prozess integriert.

Mandantenfähigkeit Als grundlegende Voraussetzung für den Testbetrieb war die zeitnahe Implementierung der Mandantenfähigkeit unabdingbar. Hierzu wurde die bisherige, sehr rudimentäre Benutzerverwaltung aus dem Prototypen zunächst in ein Modul refaktorisiert und anschließend umfassend angepasst, um eine einheitliche graphische Oberfläche zur Verwaltung verschiedener koexistierender Mandaten zu ermöglichen. Seitens der API wurden *Abteilungen* implementiert, denen andere Ressourcen - Nutzer, Sensoren usw. - zugeordnet werden können. Eine jede solche Abteilung repräsentiert nun ein Ressort, so dass alle Benutzer des Systems nur die Ressourcen einsehen können, für die sie laut der Mandantenverwaltung auch zuständig sind. Somit wird sichergestellt, dass beispielsweise Administratoren eines Ressorts ausschließlich ihre eigenen Sensoren verwalten und eigenen Vorfälle einsehen können, zudem aber auch komplexere Benutzer, die Rechte in mehreren Ressorts zugleich besitzen, angelegt werden können.

Die Umsetzung dieser Funktionalitäten umfasste Änderungen an fast allen Modulen der Webanwendung, da individuelle Filterfunktionen zur Auswahl der aktiven Beteiligung an verschiedenen Stellen der Verwaltungsanwendung benötigt wurden. Die bis zu diesem Zeitpunkt realisierte Mandantenfähigkeit besitzt zusätzlich zu den genannten Punkten auch die Möglichkeit, pro Abteilung sog. *Kontakte* zu definieren, die im Falle kritischer Ereignisse oder in regelmäßigen Abständen über den Systemstatus per E-Mail informiert werden können. Eine ähnliche Funktionalität existierte in rudimentärer Form bereits im Prototypen, wurde nun aber um eine ansprechendere Benutzeroberfläche und die Auswahl, beliebige E-Mail-Adressen oder bestehende Benutzer verwenden zu können, erweitert. Um die letztgenannte Option zu ermöglichen, war es weiterhin erforderlich, bestehende Benutzerdaten um eine E-Mail-Adresse zu ergänzen. Diese Arbeiten wurden bei der Überarbeitung der Benutzerverwaltung ebenfalls berücksichtigt.

Plattformunabhängigkeit Um in Zukunft die Auswahl der als Sensoren in Frage kommenden Hardwareplattformen zu erweitern, wurde damit begonnen, die BeagleBone-spezifischen Programmteile zu isolieren und eine Entwicklungsplattform zum Erstellen neuer Firmware aus den Debian-Paketuellen zu schaffen. Hierbei stand zunächst der Versuch im Vordergrund, die BeagleBone-Geräte auf handelsüblicher x86_64-Hardware mit Hilfe des Emulators *Qemu* vollständig zu simulieren. Mangels Unterstützung für diese Plattform wurde dieser Versuch jedoch zugunsten der Verwendung der offiziellen Buildskripte des BeagleBone-Projektes aufgegeben.

Benutzerführung und optisches Erscheinungsbild Im Rahmen der zuvor beschriebenen Refaktorisierung wurden die neuen Module mit einer auf Mobilgeräten leichter zu bedienenden Benutzeroberfläche ausgestattet. Dies umfasst sowohl die Größe und Anordnung von Bedienelementen, als auch diverse Animationen, die bei der Interaktion mit der Software die Benutzerführung leichter visualisieren und somit vereinfachen sollen. Eine wichtige Maßnahme war in diesem Zusammenhang das Entfernen vieler modaler Dialoge, die auf Mobilgeräten mit kleinem Bildschirm im Prototypen Probleme bereiteten.

Systemsicherheit Da die Änderungen im Laufe dieser ersten Iteration des Forschungsprojektes insbesondere die Serveranwendung betrafen, die zugleich die kritischste Komponente im Gesamtsystem darstellt, wurde die sowohl für alle Sensoren als auch die Benutzer erreichbare REST-API durch systematische Untersuchungen des Programmcodes auf mögliche Schwachstellen abgesucht. Dies betraf zunächst die Untersuchung aller erreichbaren Ressourcen, der darauf anwendbaren HTTP-Methoden (`GET`,

`POST`, `UPDATE`, `DELETE`), sowie der von diesen akzeptierten URL- und POST-Parameter. Weiterhin wurde sichergestellt, dass Operationen nur von dazu berechtigten Benutzern ausgeführt werden können. Analoge Überprüfungen erfolgten ebenso an der die API nutzenden Webanwendung, der aber per Definition vom Server nicht vertraut werden kann, da sie im Browser des Benutzers und somit auch eines potentiellen Angreifers ausgeführt wird. Die Integrität der serverseitigen API gebührt folglich höchste Priorität.

4 Ausblick

Die nächsten Schritte im Forschungsprojekt sind eine Konsequenz aus den zuvor beschriebenen Anforderungen und bereits durchgeführten Arbeiten. Gemäß der gesetzten Prioritäten ist das Whitelisting, bzw. die Implementierung einer generischen Filterfunktion für eintreffende Ereignisse von großer Bedeutung für die zukünftige Errichtung einer umfangreichen Testinstallation innerhalb des Produktivnetzwerkes. Die Konzeption und Entwicklung des Dienstes zum Sammeln zusätzlicher Informationen über Verbindungsversuche ist anschließend ebenfalls von Interesse, während die Portierung der Sensorsoftware auf weitere Plattformen aufgrund der inzwischen nun doch verhältnismäßig leicht zu beschaffenden BeagleBone-Hardware eher geringe Priorität genießt, jedoch insbesondere zur Einrichtung größerer Testinstallationen mit bereits bestehender Hardware von Vorteil sein sollte. Die Verbesserung der Systemsicherheit, die Refaktorisierung und Modularisierung, sowie die schrittweise Anpassung der Webanwendung zur Verbesserung des Benutzererlebnisses sind hingegen langanhaltende, konstante Prozesse.

5 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 untermauern.

Revision	Änderungen
205	Cleanup of old Aptana/Eclipse preference files
206	Added PSR-4 compliant autoloader so that the name of the root directory can now be arbitrary instead of 'HoneySens' - Added simple DB setup process in case the 'user' table is missing (this should be made interactive in the future) - Bootstrap functions now can add (error) messages to an array that is available to all routes
207	Major refactoring of the server application structure. Included grunt as build system and require.js for dynamic dependency-based loading of javascript code. The JS portion of the app was also split into several files and made modular. Dockerfile updated.
208	Some bug fixes related to the previous refactoring and integration of the beanstalk scripts into the build process
209	Proper password field within the user management GUI
210	Doctrine CLI script paths repaired
211	Division model added on the backend, modal.js -> regions.js as a plugin containing reusable regions
212	Marionette framework updated to version 2.4.1, dependent code updated to work with that version using the official updater script and some common sense. Might still contain some errors.
213	- Added division model for user/sensor grouping - Added the ünderscore-tpl"javascript library for template separation into individual files - Submodule support added (with own controller, views and templates) - Basic account management realized as such a separate module
214	Added basic 'watch' task to the Gruntfile that only builds changed files
215	- Allow addition/removal of users to/from divisions - Some smaller fixes
216	Fixed generic routing module to properly work with multiple routes per module

217	- Restructured 'accounts' module so that all internal module routing and command logic is inside the module.js file and not within some arbitrary view class. This enables the fast definition of URL routes for all the existing functionality. Also, the module start() method instantiates the base view that is required for all the routing functions. - Replaced the Marionette.wreqr commands interface with the request/response interface, because in their implementation commands are stored when there's no fitting command handler and executed later, when such a handler becomes available. This leads to problems with modules that define their handlers only at instantiation. This may be solved in a better way later by switching to the Marionette.radio interface.
218	Restructured the controller of the 'accounts' module. The browser URL bar is now properly updated on page changes.
219	Converted sensor views to the 'sensor' module with division support. Also includes some smaller layout fixes. Added "*.tpl"filetype to .htaccess so that templates are sent with the correct Content-Type header.
220	Restructured the settings view (UI-wise) to match new design rules
221	- API methods added to retrieve events according to various parameters. Bootstrapping changed to only return events that the current session user is allowed to view. - Various UI fixes when displaying events
222	Controller API changed to process all data retrieval requests within a generic get($attributes) method. Events are now implemented like that. It allows the combination of multiple request attributes without much hassle - for example to only return data that belongs to a certain user or division. Also switched from full DQL strings to using the DQL builder for query creation.
223	Frontend division models are now synchronized correctly during login and logout. Fixes an issue where divisions weren't available after login.
224	Division and sensor controllers transformed to the new get($attributes) API, so that only the relevant entities for the currently logged in user are provided by the server. Frontend simplified accordingly.
225	Each contact now belongs to a specific division and Users have an additional E-Mail attribute. System settings implemented as own module (currently only providing SMTP configuration).