Technische Universität Dresden

Fakultät Informatik

Professur für Datenschutz und Datensicherheit

Projekt HoneySens

# 3. Zwischenbericht

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

Stand: 16.01.2016

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. Ein weiterer wichtiger Bestandteil des Forschungsprojektes ist zudem die Koordinierung der vielfältigen Anforderungen der am Projekt interessierten und teilnehmenden Ressorts der Landesverwaltung.

Der zweite Zwischenbericht beinhaltete eine Reihe funktionaler Erweiterungen und die Einführung eines sauberen Build- und Installationsprozesses für die verschiedenen Softwarekomponenten. In der dritten Iteration des Projektes hingegen lag nun der Fokus insbesondere auf dem Bereich der Qualitätssicherung und der Vorbereitung der bevorstehenden Testsysteme innerhalb verschiedener interessierter Institutionen. Auch für den testweisen Einsatz in produktiven Netzen ist es unabdingbar, das Gesamtsystem ausführlich hinsichtlich Funktionalität und Sicherheit zu überprüfen. Das Auffinden und Beheben von Bugs ist dabei ein anhaltender Prozess, der den nachfolgenden Testeinsatz konstant begleiten wird.

Beim Abgleich der Anforderungen für die Implementierung des Testsystems innerhalb des Sächsischen Verwaltungsnetzwerkes mit den ursprünglich getroffenen Annahmen der Diplomarbeit stellte sich heraus, dass noch einige Nacharbeiten erforderlich sein werden, um tatsächlich zuverlässige Konnektivität zwischen Sensoren und Server im realen Netzwerk herstellen zu können. HTTP und HTTPS können tatsächlich nicht über Gateway-Grenzen und Firewalls hinweg genutzt werden, sondern erfordern die Verwendung allgegenwärtiger Proxy-Server. Die Implementierung für deren Unterstützung hatte in dieser Phase oberste Priorität.

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 16. Oktober 2015 bis zum 16. Januar 2016 durchgeführten Arbeiten am HoneySens-Projekt. Teilbereiche, die in den Anforderungen des ersten Projektberichtes genannt, aber hier nicht weiter beschrieben werden, sind Gegenstand der zukünftigen Weiterentwicklung.

**Sensor-Verwaltung** Die zuvor genannten neue Anforderung, HTTP(S)-Proxy-Support in die Architektur zu integrieren, führte letztendlich zu einer Überarbeitung der gesamten Sensorverwaltung, da in diesem Zusammenhang auch direkt mit Altlasten aus der Diplomarbeits-Prototypen-Phase aufgeräumt und

einige einige Features mit geringer Priorität ohne nennenswerten Mehraufwand integriert werden konnten. Dies umfasste insbesondere die Spezifikation beliebiger MAC-Adressen für die Netzwerkschnittstelle der Sensoren (oder die Nutzung der gerätespezifischen Standardadresse), sowie eine dynamische Definition des Server-Endpunktes, also der Adresse, unter der ein Sensor den Server erreichen kann. Bisher handelte es sich dabei um eine nach der initialen Einrichtung nicht mehr veränderliche URL. Nach den Änderungen ist es nun hingegen möglich, entweder die globale Standardadresse des Servers (aufgeteilt in Hostname, Port für die Webanwendung (HTTPS) und Port für den Zeitabgleich (HTTP)) zu übernehmen oder diese Daten sensorspezifisch einzustellen. Alle genannten Parameter zur Sensorkonfiguration können nun zudem auch nach dem Registrieren eines Sensors noch modifiziert werden. Somit ist auch eine Änderung der Netzwerkkonfiguration im laufenden Betrieb möglich.

Der Dialog zur Sensorkonfiguration in der graphischen Verwaltungsoberfläche bot für all diese Optionen nur verhältnismäßig wenig Platz und musste deswegen einem Panel weichen, das sich vom rechten Bildschirmrand aus in den Sichtbereich des Benutzers schiebt (analog zu von mobilen Betriebssystemen bekannten UI-Patterns). Dieses bietet erheblich mehr Raum zur Gestaltung und ist nebenbei auf Mobilgeräten komfortabler nutzbar als modale Dialoge. Die Optionen zur Konfiguration selbst wurden stark modifiziert und nutzen jetzt einheitliche *Radio Buttons*, um für jede Konfigurations-Kategorie auszuwählen, ob der Standardwert - also beispielsweise Nutzung eines DHCP-Servers, die gerätespezifische MAC-Adresse oder kein Proxy - genutzt oder stattdessen individuelle Parameter vergeben werden sollen. Der Button zur Erzeugung des Sensor-Schlüssels und -zertifikats aus dem alten Konfigurationsdialog wurde zudem entfernt, da dieser Prozess für den Nutzer völlig transparent vom Server erledigt wird.

Eine weitere serverseitige Neuerung ist, dass nach Abwägung potentieller Risiken die für Sensoren generierten Schlüssel nun zusätzlich noch auf dem Server vorgehalten werden. Somit lässt sich das Konfigurationsarchiv für einen Sensor nach Bedarf jederzeit neu erzeugen. Dies wird beispielsweise notwendig, wenn ein Sensor aufgrund eines Hardwaredefektes ausfällt und neu initialisiert werden muss.

**Sensor-Software** Nach den zuvor genannten Änderungen auf der Serverseite war es erforderlich, auch die Clients, sprich die auf den Sensoren laufende Software, entsprechend anzupassen. Dies wurde zum Anlass genommen, auch an dieser Stelle Altlasten zu entfernen und zu homogenisieren: Statt der in verschiedensten Sprachen geschriebenen Skriptsammlung werden die Kernaufgaben eines jeden Sensors nun von einigen wenigen Python-Skripten übernommen, die wiederum ein einheitliches Interface (im Sinne der übergebenen Parameter) beim Aufruf nutzen. Zusätzliche, plattformspezifische Funktionalität kann bei Bedarf in Form weiterer Skripte hinzugefügt werden. Eine Übersicht über die Kernskripte gibt Abbildung 1.

Die Basis-Sensorfunktionalität ist folgendermaßen verteilt:

**config_sensor.py** : Erwartet als Argument den Pfad zu einem Sensor-Konfigurationsarchiv und führt anschließend die Sensorkonfiguration durch. Resultat dieses Prozesses ist (unter anderem) die zentrale Konfigurationsdatei `honeysens.cfg`, die von allen anderen Skripten genutzt wird. Das Skript wird entweder manuell vom Benutzer bei der Einrichtung eines Sensors ausgeführt oder wird - im Falle des BeagleBone Black - beim Systemstart von einem plattformspezifischen Initialisierungsskript aufgerufen, wenn eine neue Sensorkonfiguration in einem vorgegebenen Verzeichnis gefunden wurde.

**poll.py** : Führt das periodische Polling aus, mit dem Sensoren ihren Systemstatus zum Server senden und

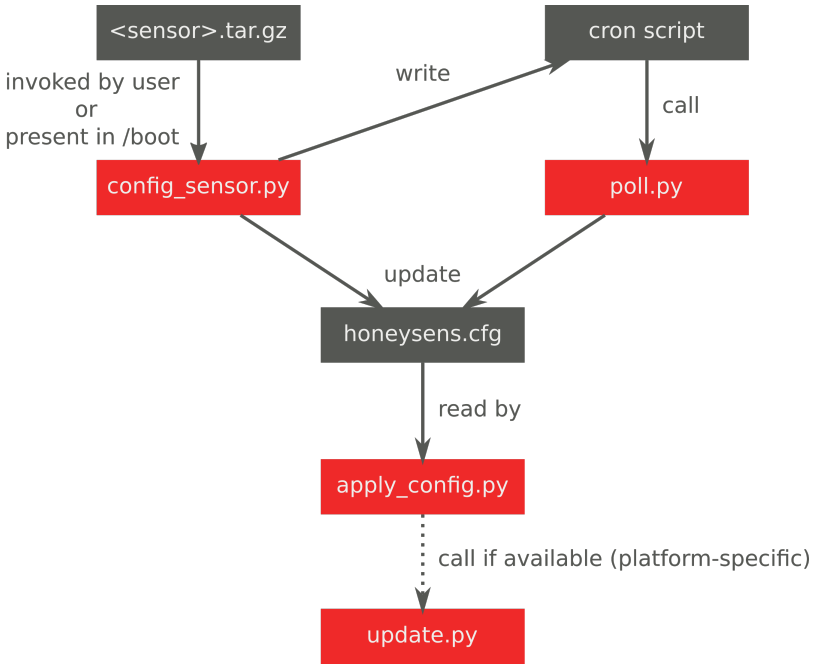


Abbildung 1: Architektur der Sensor-Software

die aktuelle Konfiguration empfangen. Spiegelt die Sensorkonfiguration in der Datei `honeysens.cfg` wieder und synchronisiert diese mit dem Server. Das Polling-Skript wird automatisch durch einen *Cron-Job* aufgerufen.

**apply_config.py** : Liest die Sensorkonfiguration aus `honeysens.cfg` aus und passt den aktuellen Systemzustand gemäß dieser an. Dies beinhaltet u.a. das Starten und Beenden der Honeypot-Dienste, die Aktualisierung der Netzwerkeinstellungen und das Anstoßen eines Update-Vorganges, falls notwendig.

**update.py** : Dieses Skript enthält Update-Logik, wenn die entsprechende Sensorplattform (siehe nachfolgender Abschnitt) Softwareupdates unterstützt. Es bezieht ein Abbild der zu aktualisierenden Firmware vom Server und führt dieses aus.

Das Python-Skript zur serverseitigen Erzeugung eines neuen Konfigurationsarchives für Sensoren wurde bisher direkt von der Webanwendung aufgerufen, was im Falle einer hohen Serverlast zu Timeouts führen könnte. Aus diesem Grund wurde ein Beanstalk-Worker geschrieben, der die bereits vorhandene Warteschlangen/Job-Infrastruktur nutzt und Konfigurationen mit allen neu hinzugefügten Parametern erstellen kann. Im gleichen Schritt wurde dessen Abhängigkeit von externen Template-Dateien aufgelöst: Die Konfiguration wird nun direkt vom Job-Skript geschrieben.

**Plattformunabhängigkeit** Als Fortsetzung der im zweiten Zwischenbericht beschriebenen Arbeiten zur Plattformunabhängigkeit wurden im Rahmen der neu entwickelten Sensorsoftware alle für den *BeagleBone Black* spezifisch geschriebenen Komponenten separiert und zu einem eigenen Paket zusammengefasst, welches ebenfalls über das Debian-Paketmanagement installiert werden kann. Als Resultat enthält das DEB-Paket `honeysens-sensor` nun alle plattformunabhängigen (aber architekturabhängigen) Bestandteile, während `honeysens-sensor-platform-bbb` alle jene Skripte enthält, die für

die BeagleBone-Plattform und die nur darauf zur Verfügung stehende Funktionalität (also insbesondere atomare Systemupdates) benötigt werden.

Weiterhin wurde eine entscheidende Änderung ganz im Sinne der Zuverlässigkeit der BeagleBone-Sensoren eingebracht: Die Geräte laufen nun standardmäßig vom internen (*eMMC*-)Speicher und nutzen die externe microSD-Karte nur noch als Zwischenspeicher während des Updateprozesses und, wie bisher, zur Neuinstallation eines Gerätes. Der Updateprozess wurde hierfür stark modifiziert und nutzt dafür auch die offiziell vom BeagleBone-Projekt bereitgestellten Skripte zur Installation und Initialisierung eines Gerätes. Längerfristig verspricht dies eine Steigerung der erwarteten Lebensdauer der Geräte, da der interne Speicher als zuverlässiger gilt als handelsübliche microSD-Karten. Als Nebeneffekt der Modifikation können Sensorupdates nun auch deutlich schneller durchgeführt werden.

Für die Installation und Aktualisierung der Sensoren wird Firmware benötigt, die neben den zuvor bereits erwähnten Debian-Paketen mit der HoneySens-Software auch noch das umliegende Debian-Basissystem in einer Form beinhalten, die für den jeweiligen Vorgang (Installation/Update) genutzt werden kann. Bisher wurden hierfür gesonderte Versionen benötigt: Die „kanonische“, vom Server akzeptierte Variante der Firmware konnte nur für Updates direkt verwendet und musste zunächst zu einem microSD-Image umkonvertiert werden, um auch eine Installation von Neugeräten zu ermöglichen. Diese Zweiteilung wurde nun aufgehoben: Der neue Updateprozess ermöglicht es, für all diese Operationen ein einheitliches Abbild zu verwenden und den Konvertierungsprozess zu entfernen. Als Resultat dieser Änderungen konnte einerseits die Komplexität des Updatevorgang verringert (und folglich die Fehlerwahrscheinlichkeit gesenkt) und andererseits auch die Anforderungen an den Server hinsichtlich Speicherplatz und Rechenleistung (der Konvertierungsprozess konnte je nach System durchaus 10 Minuten andauern) vermindert werden. Das Generieren der Firmware wird wie üblich von einem Skript übernommen, das alle notwendigen Schritte vom Download einer aktuellen, stabilen Debian-Plattform bis zur Integration und Konfiguration der Debian-HoneySens-Pakete vollautomatisch übernimmt.

**Deployment** Der auf *docker* basierende Deployment-Prozess wurde angepasst und nutzt nun sogenannte *Volumes* ergänzt, auf denen die persistenten Daten eines HoneySens-Servers (dessen Konfiguration, Inhalt der Datenbank, hochgeladene Firmware-Versionen usw.) abgelegt werden. Es ist dadurch zudem den Administratoren möglich, beim Installation eines Servers sogenannte *Data-only container* zum Vorhalten der Daten zu erstellen. Es handelt sich dabei um ein Pattern zum Management für Docker-Container, das bei Container-Updates sicherstellt, dass die persistenten Daten nicht verloren gehen. Als Resultat der Änderung ist ein Austausch eines HoneySens-Server-Containers durch eine neuere Version im laufenden Betrieb ohne zusätzlichen administrativen Aufwand möglich.

Zusätzlich wird die gesamte Sensor-Software für den praktischen Testeinsatz nun im *Release*-Modus gebaut, in dem die Bestandteile des Web-Frontends konkateniert und komprimiert (*„minified“*) werden, um die beim Aufruf der Website nötigen HTTP(S)-Anfragen auf ein Minimum zu reduzieren, was sowohl Server als auch Clients entlastet.

**Testbetrieb** Parallel zu den genannten Modifikationen der Software wurde der Testbetrieb angestoßen. Ein als Docker-Plattform geeigneter Server innerhalb des SVN wurde organisiert, gehärtet und mit der Serversoftware versehen. Der gewählte Deployment-Prozess erwies sich hier als ausgesprochen effektiv und führte zu keinen unvorhergesehenen Problemen. Die Integration von Sensoren gestaltete sich jedoch

schwierig, da entgegen der bisherigen Annahmen keine direkte Route in Servernetze existiert und eine Nutzung von HTTP(S)-Proxy-Server zwingend notwendig ist. Nachdem diese Tatsache nach einigen Tests vor Ort feststand, wurden die zuvor genannten Änderungen eingebracht. Im Laufe weiterer Tests im Januar 2016 führten weitere organisatorische Verzögerungen und ein Fehler der Sensoren beim Umgang mit TLS-Zertifikatsketten dazu, dass der Betrieb der Sensoren bis zur Fertigstellung dieses Berichts noch nicht beginnen konnte. Es ist jedoch davon auszugehen, dass spätestens im Februar 2016 erste Sensoren offiziell in den Dauerbetrieb übergehen können.

Für einen Testbetrieb innerhalb des Campusnetzes der TU Dresden haben sich mehrere interessierte Administratoren gemeldet, womit mehrere Fakultäten und Struktureinheiten der Universität am Test teilnehmen werden. Die organisatorischen Rahmenbedingungen werden gerade in Absprache mit der *Stabsstelle Informationssicherheit* der TU geschaffen und auch mit einem Artikel im hiesigen *Universitätsjournal* konnte das Projekt auf sich aufmerksam machen. Es ist hier ebenfalls damit zu rechnen, im Laufe der nächsten Wochen in einen geregelten Testbetrieb übergehen zu können.

**Sonstiges** Zuletzt wurden unzählige Bugs in verschiedenen Komponenten des Systems behoben, die während des Testbetriebes ans Tageslicht getreten sind. Die Commit-Auflistung im Anhang gibt einen groben Überblick über alle behobenen Probleme. Um mit den während der Testphase zu erwartenden Bug-Reports und Wünschen der Nutzer strukturiert umgehen zu können, wurde nach Evaluation verschiedener Tools außerdem eine webbasierte Software namens *„Phabricator“* zum internen Projektmanagement aufgesetzt.

# 3 Ausblick

Die Voraussetzungen für den Testbetrieb seitens der Software sind erfüllt, weshalb die nächste Projektphase vor allem organisatorische Arbeiten mit sich bringen wird. Dies umfasst insbesondere die Schulung von Administratoren, die das System in Zukunft nutzen wollen. Es ist zudem damit zu rechnen, dass weiterhin Fehlerberichte und neue Anforderungen aus dem Testbetrieb hervorgehen werden, die gemäß ihrer Dringlichkeit der Reihe nach umgesetzt werden müssen. Es gibt zudem auch noch offene Anforderungen geringerer Priorität, die während der ersten drei Projektphasen hinzugekommen sind. Ausführlichere Penetrationstest zur Überprüfung der Sicherheit der Architektur, sowie Tests bezüglich der Skalierbarkeit und des Verhaltens unter Last sind außerdem weiterhin von Interesse.

# 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 |
|---|---|
| 281 | Fixed a bug where the recon service wasn't started correctly on sensors |
| 288 | - Sensor deb build process updated<br>- BBB specific build script added (not final) |
| 302 | - Sensor configuration dialogue („add"/„remove") fully rewritten to support network setting persistence, custom MAC and proxy settings<br>- Sensor software fully rewritten in python and reduced to three major scripts: polling, install_config and apply_config. All BeagleBone specific parts moved to the platform directory.<br>- App settings GUI adjusted to match the new sensor configuration options (mostly regarding the server endpoint config)<br>- Slideable overlay regions introduced to the web GUI as an replacement for modal dialogues<br>- Sensor configuration archive beanstalk worker clean rewrite to support all new sensor settings. Also pulls all relevant information from the app config and doesn't rely on template files anymore. |
| 303 | - apply_config.sh can now only be run once simultaneously<br>- Fixed a bug that pevented poll.py to be run at system startup<br>- Preparations for firmware update<br>- Current firmware version is now part of the sensor sw distribution (version.txt) and the configured firmware according to the sensor config is part of honeysens.cfg and distributed during polling and sensor creation |
| 304 | BeagleBone notes updated |
| 305 | Platform specific sensor scripts for the BeagleBone Black consolidated and rewritten. Skeleton for a separate debian package for those components added. |
| 306 | - Added API endpoint that allows sensors to retrieve their firmware directly via their sensor id<br>- Visibility of sidebar items is now easier to control by also taking into account the 'action' of permission domains<br>- Fixed a bug that caused an error when the sensor configuration overview is shown after a second login |
| 307 | Fixed a few missing init paths that led to incorrect sensor behaviour |
| 308 | Fixed a bug that prevented event details to be shown for users without admin status |

| | |
|---|---|
| 309 | - Sensor SSH daemon restricted to only start when the USB ethernet gadget of BeagleBones is brought up<br>- Firmware controller code cleaned up a bit and validation checks adapted to new firmware layout<br>- Sensors now correctly report their software version instead of a dummy value |
| 310 | Sensor firmware raised to version 0.1.0. Full system updates of the BBB platform are now supported again and a lot of bugs were fixed on the sensor side. |
| 311 | - Removed the API endpoint for SD card images and associated controller code<br>- Removed SD card specific features from the firmware GUI<br>- Added API endpoint to download previously uploaded firmware |
| 312 | Docker build scripts updated, containing the new sensor config creation worker and basic volume support |
| 313 | - The polling script now only restarts network connections when the configuration actually changed<br>- Fixed a recon service memory leak (scapy by default stores all sniffed packets)<br>- Correct handling of multiple polling processes running in parallel, which is also used to give feedback about running updates<br>- Sensor cron script does now nothing until the sensor is configured<br>- BBB: Systemd unit files added to ensure that the sensor SSHD starts after the USB ethernet gadget is configured<br>- BBB: Polling starts directly after the initial polling procedure<br>- BBB: Fixed a bug where the update failed because we didn't use absolute path to some binaries |
| 314 | - BBB: Fixed a bug that prevented SSHD to start after a sensor was configured<br>- BBB: Added SSH and NTP configuration to the platform-specific package |
| 315 | - passiveScan globally changed to recon<br>- Sensors keys are now saved within the database and used to regenerate sensor configurations on demand<br>- Sensor add/edit form validation finalized<br>- The beanstalk sensor config creation worker now starts a new mysql connection for each job to prevent timeouts when using only a single one<br>- Fixed the docker build process: Volumes are now added after a new container has been built |
| 316 | HTTP(S) proxy support added to the sensor scripts |
| 317 | - Server configuration moved to the /data directory so that it's part of the docker data volume<br>- Server version bumped to 0.1.0-rc4 |
| 318 | Fixed a bug where the htpdate config wasn't updated under some circumstances |

| | |
|---|---|
| 319 | - Fixed some permissions, so that managers can't change global settings or firmware-related stuff<br>- Fixed a minor bug that prevented logged-in users to GET the default sensorconfig directly |
| 320 | - Fixed a bug where the sensor config creation worker didn't start within a docker container due to a misleading app config path<br>- Server version bump to 0.1.0 (from rc4) |
| 321 | Platform-specific modifications for the BBB build script |
| 322 | Some updates for the BBB notes |
| 328 | Added loading GIF, which was missing |
| 329 | - Fixed a bug where the polling script sent sensor data with invalid timestamps if the system date had to be changed during its run<br>- Timestamp validity checks re-enabled on the server side for the polling process (as a follow-up to the previous fix)<br>- Default encoding of the beanstalk worker hardcoded to UTF-8 to ensure proper string handling |
| 331 | - Version bump to 0.1.1<br>- Fixed a bug that caused the sensor config creation worker to fail on startup (caused by the previous introduced default encoding)<br>- Docker setup script updated in terms of the moved app config file location |
| 332 | Fixed a bug that caused the minified release version of the web UI to throw an error after login (but only on some browsers). The cause seems to be the jquery.piechartcountdown plugin, which tries to dynamically insert some CSS rules prior to the existing ones. This works when the CSS files are separate (dev build), but may lead to problems when the CSS files are concatenated and minified for release, because the then present @charset and @include lines always have to be the first CSS statements. For now, the only @charset definition (included by jquery.fileupload.js) was removed to avoid that error. |
| 333 | Server-side implementation of gateway and DNS support for sensors |
| 334 | - Sensor-side implementation of gateway and DNS support<br>- Sensor software version raised to 0.1.3 for all components (including the platform-specific ones). Those should stay in sync in the future.<br>- Web UI: Gateway and DNS fields are now optional |
| 335 | - Server version bumped to 0.1.2<br>- Apache config updated with a redirect to HTTPS (for HTTP) and SSL settings from bettercrypto.org |