Technische Universität Dresden

Fakultät Informatik

Professur für Datenschutz und Datensicherheit

Projekt HoneySens

4. Zwischenbericht

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

Stand: 16.04.2016

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.

In der vorangegangenen dritten Projektphase lag der Fokus auf dem Bereich der Qualitätssicherung und der Vorbereitung der bevorstehenden Testsysteme innerhalb der teilnehmenden Institutionen, sowohl innerhalb des Sächsischen Verwaltungsnetzes als auch an der TU Dresden. Ein wesentlicher Punkt des dritten Berichtes war die Implementierung einer breiten Unterstützung der Sensor-Server-Kommunikation über HTTP- und HTTPS-Proxy-Server hinweg. Diese Arbeiten wurden bereits in der dritten Projektphase so weit durchgeführt, dass eine Kommunikation über Proxy-Server hinweg im Labor problemlos möglich war. Beim Ausrollen der Sensoren in der Sächsischen Landesverwaltung stellte sich jedoch heraus, dass für die Kommunikation mit den dort vorhandenen Proxy-Servern weitere umfangreiche Änderungen notwendig sind. Der reibungslose Betrieb des Systems in Zusammenspiel mit verschiedensten Web-Proxies war zentraler Bestandteil dieser Projektphase. Alle nachfolgend genannten Änderungen fanden mit Blick auf das übergeordnete Ziel statt, den Testbetrieb so schnell wie möglich voranzutreiben und folglich die Softwarebasis entsprechend zu stabilisieren.

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 17. Januar 2016 bis zum 16. April 2016 verzeichneten Fortschritte am HoneySens-Projekt. Teilbereiche, die in den Anforderungen des ersten Projektberichtes genannt, aber hier nicht weiter beschrieben werden, sind Gegenstand der zukünftigen Weiterentwicklung.

Sensor-Server-Kommunikation Das Zusammenspiel der Sensorsoftware mit den im SVN genutzten Proxy-Servern stellte sich als schwierig heraus, nachdem die Tests im Labor mit dem freien Proxy-Server *squid* zunächst erfolgreich verlaufen waren. Grund hierfür war gleich eine Reihe von Faktoren: Im SVN ist bisweilen eine Authentifizierung zur Nutzung der Proxy-Server erforderlich, was eine Anforderung ist, die ursprünglich nicht bekannt war. Hinzu kommt, dass für die Authentifizierung verschiedene standardisierte Verfahren existieren und auch mehrere von diesen unterstützt werden sollten, da die Server von ihren jeweils eigenen Verantwortlichen administriert werden und es nicht vorhersehbar ist, welche Methoden pro Teilnetz unterstützt werden. Um die Sensor-Administration weiterhin möglichst einfach zu halten, wurde gemeinsam mit diesen Erkenntnissen noch die Anforderung gestellt, die pro Sensor geforderte Authentifizierungsmethode automatisch erkennen zu können (*Autodiscovery*). Web-Proxy-Server liefern bei jeder Anfrage eine Liste aller von ihnen unterstützen Authentifizierungsmethoden aus, die vom Client interpretiert und für nachfolgende Anfragen genutzt werden kann.

Ein weiteres Problem war, dass die Python-Bibliothek *python-requests*, die von den Sensoren zur Kommunikation mit dem Server genutzt wurde, nur eine sehr rudimentäre Unterstützung für Authentifizierungsverfahren an Proxies bietet. Es wurde deshalb zunächst damit begonnen, python-requests um die fehlende Funktionalität zu ergänzen. Entsprechende Vorarbeiten waren auch bereits als Teil anderer Projekte auf *github* zu finden. Leider stellte sich bei deren Integration schnell heraus, dass der grundlegende Aufbau der requests-Bibliothek eine Implementierung der *Autodiscovery*-Funktion unmöglich machte. Grund dafür war, dass die Bibliothek wiederum auf dem Paket *urllib3* aufbaut, das eine Fehlerbehandlung im Falle einer Proxy-Fehlermeldung nicht vorsah. Da der Autodiscovery-Mechanismus aber auf eine solche Testnachricht zum Abfragen der unterstützen Authentifikationsverfahren angewiesen ist, wurde letztlich nach vollwertigen Alternativen zur requests-Bibliothek gesucht.

Letztendlich fiel die Wahl auf die beliebte *libcurl*, bzw. deren ebenfalls quelloffene Python-Anbindung *PycURL*. Die Bibliothek ist Grundbestandteil jeder Linux-Distribution und bietet auch eine umfassende Unterstützung für alle gängigen Proxy-Authentifikationsverfahren, darunter Basic-, Digest-, NTLM- und Kerberos-Authentifizierung. Um für zukünftige Änderungen die Abhängigkeiten von der externen Bibliothek möglichst gering zu halten, wurden alle kommunikationsbezogenen Methoden, die nun von PycURL abhängen, in ein eigenes Python-Modul ausgelagert. Von allen anderen Skripten, die mit dem Server kommunizieren möchten, kann so die benötigte Funktionalität leicht zusätzlich eingebunden werden. PycURL bringt bereits eine eigene Autodiscovery-Funktionalität mit. Diese funktioniert mit dem Labor-Testproxy hervorragend, verursachte allerdings vor Ort mit einem Test-Proxy im SVN erneut Probleme. Es stellte sich heraus, dass die für den Sensor eingerichteten Zugangsdaten lediglich über eine NTLM-Authentifizierung nutzbar waren, nicht jedoch über Kerberos. Da der Proxy aber beide Methoden öffentlich anbot, entschied sich cURL für die „falsche“ Variante und ein Verbindungsversuch schlug fehl. Als Ausweg wurde schließlich händisch noch eine Brute-Force-Detektionsroutine hinzugefügt, die in einem solchen Fall alle bekannten (sicheren) Authentifizierungsmethoden blind durchtestet. Sobald eine Nutzung des Proxies erfolgreich ist, wird die korrekte Methode in einem lokalen Cache gespeichert und

vom Sensor für alle zukünftigen Anfragen genutzt. Falls in Zukunft die gewählte Methode nicht mehr unterstützt werden sollte, wird nach einem Fehlschlag bei der Authentifizierung erneut die Detektionsroutine gestartet. Nach all diesen Änderungen verlief die Kommunikation zwischen Sensor und Server in allen bisher getesteten Szenarien reibungslos.

Ein weiterer Konflikt entstand nach dem Ersetzen von python-requests durch PycURL bei der automatischen Zeitsynchronisation. Da diese ebenfalls über HTTP ablief, war auch hier die Kommunikation über Proxy-Server erforderlich. Leider bot der dafür verwendete Dienst, *htpdate*, keine Unterstützung für Proxy-Authentifizierungsverfahren an. Folglich musste das in C geschriebene Tool entweder um diese Funktionalität erweitert oder durch einen anderen, besseren Mechanismus ersetzt werden. Da die automatische Verwaltung der htpdate-Konfigurationsdateien und deren Synchronisation mit der serverseitigen Sensorkonfiguration verhältnismäßig komplex war, erschien letztgenannte Methode interessanter. In der Folge wurde der htpdate-Dienst vollständig entfernt und dessen Funktionalität in das Polling-Skript eingearbeitet. Somit verwenden nun auch diese Abfragen die einheitlichen, auf PycURL aufsetzenden Methoden zur Kommunikation mit dem HoneySens-Server. Ein weiterer bedeutender Vorteil dieser Änderung ist außerdem, dass auch die Zeitsynchronisation nun über HTTPS (TCP-Port 443) erfolgen kann und unverschlüsselte Kommunikation über Port 80 prinzipiell nicht mehr stattfindet.

Im Zuge der genannten Erweiterungen wurden zudem auch noch Änderungen an der graphischen Benutzerschnittstelle fällig: Felder für den Proxy-Benutzernamen und das Passwort wurden hinzugefügt. Ebenso stellte sich bei einem Test in einem Netz ohne DHCP heraus, dass DNS-Server und Gateway zur Kommunikation mit einem über einen Hostnamen spezifizierten Server zwingend benötigt wurden. Diesen Umstand wurde jetzt Rechnung getragen, indem die Parameter nun auch über die „manuelle Netzwerkkonfiguration“ eingetragen werden können.

Updateprozedur der Serversoftware Da sich in Vorbereitung der Testläufe bereits mehrere HoneySens-Server im Einsatz befanden, wurden strukturierte Verfahren für deren Softwareaktualisierung nötig. Große Teile dieser Aufgabe werden bereits vom Docker-Deployment-Prozess übernommen. Die Daten jedoch, die außerhalb eines solchen Containers liegen, benötigen unter Umständen ebenfalls Modifikationen. Dies betrifft insbesondere das verwendete Datenbankschema, das von Version zu Version hin aktualisiert werden muss. Da zunächst vermieden werden sollte, Schema-Updates über gesonderte SQL-Skripte auszuliefern, wurde auf ein Feature des verwendeten Datenbankabstraktionsframeworks *doctrine* zurückgegriffen. Dieses kann Differenzen zwischen zwei Schemata (also beispielsweise des neuen und des gerade genutzten Schemas) berechnen und im Falle der Konfliktfreiheit auch eine Aktualisierung herbeiführen. Diese Methode funktionierte in Tests problemlos und steht nun dem Administrator in der Systemverwaltung zur Verfügung. Nach jedem Update der Server-Software ist nun eine Aktualisierung des Schemas erforderlich.

Bugs, Stabilität und Sicherheit Im Rahmen des beginnenden Testbetriebs und der Nutzung des Systems durch Administratoren wurden verschiedene Probleme und Verbesserungswünsche deutlich. Ein Teil der durchgeführten Veränderungen soll die nachstehende Auflistung verdeutlichen:

- Die Standardkonfiguration des Webservers im Docker-Image wurde so modifiziert, dass Anfragen via HTTP auf HTTPS (Port 443) umgeleitet werden. Die Konfiguration des Apache-SSL-Modules wurde um einige sicherheitskritische Parameter gemäß aktuellen Empfehlungen gemäß des Ratge-

bers auf `http://bettercrypto.org/` ergänzt.
- Es traten Probleme bei der Verifikation von nicht-selbstsignierten Serverzertifikaten auf, da die Sensoren diese unter Umständen nicht selbst verifizieren konnten. Um dieses Problem zu lösen, bekommen Sensoren nun bei ihrer Konfiguration die vollständige Zertifikatskette mitgeteilt. Das Serverzertifikat kann nun komfortabel beim Erstellen des Docker-Containers über einen Kommandozeilenparameter eingebunden werden.
- Das Docker-Image setzt nun auf aktualisierter Betriebssystem-Software auf und wird insbesondere mit einer aktualisierten Version des Apache-Webservers ausgeliefert.
- Es wurden eine Reihe von Fehlern behoben, die dazu führten, dass der *recon*-Modus Antwortpakete von Proxies und systemweiten DNS-Servern als Ereignisse klassifizierte. Auch die Behandlung von Paketen ohne Payload wurde verbessert.
- Um mit den teils sehr unzuverlässigen Systemuhren der BeagleBone-Sensoren besser umgehen zu können, wurde die serverseitige Validierung der übermittelten Zeitstempel liberaler gestaltet.
- Die webbasierte Verwaltungsoberfläche hat einige Verbesserungen im Backend erfahren, damit beim Speichern von Änderungen keine Fehler mehr auftreten, weil das entsprechende Model im Hintergrund bereits vom Server wieder aktualisiert wurde. Die Synchronisation der server- und browserseitigen Models erfolgt nun effizienter.

Überarbeitung Ereignisübersicht Die Liste aller aufgezeichneten Ereignisse bezog bisher bei jedem Aufruf alle gesammelten Ereignisse vom Server, was bei einer großen Anzahl von Vorfällen, wie es insbesondere nach einem langen Deployment des Systems zu erwarten ist, zu großen Problemen führen kann. Die Interaktion mit der Liste verläuft dann nur noch sehr träge und im Extremfall ist es denkbar, dass serverseitige Speicherlimits bei der Bearbeitung einer Anfrage überschritten werden. Um auch langfristig die Stabilität des Systems sicherstellen zu können, ist folglich eine Überarbeitung der Ereignisliste notwendig geworden. In diesem Zuge wurde ebenfalls beschlossen, die bisher angesammelten Verbesserungsvorschläge direkt mit einzubeziehen. Diese betrafen primär zusätzliche Such- und Filtermöglichkeiten, sowie die Inklusion von MAC-Adressen der Senderseite eines jeden Ereignisses.

Die bisherige Implementierung setzte auf ein jQuery-Plugins namens *DataTable*, das leider auch ein separates Datenmodell verwaltet und somit die Ereignisse noch zusätzlich mittels einer Adapterschicht in das geforderte Format konvertiert werden mussten. Bei der Recherche stellten sich das Widget *Backgrid.js* und die zugehörige Plugin-Bibliothek *PageableCollection.js* als idealer Ersatz für die bisherige Lösung heraus. Beide Komponenten setzen auf der Basiskomponente *Backbone.js* auf, die auch die Grundlage des browserseitigen HoneySens-Frontends bildet. Die Implementierung nutzt nun eine nneue serverseitige API-Schnittstelle, über die kleinere Listen von Ereignissen (beispielsweise zur Darstellung einer einzelnen Seite) abgerufen werden können. Die Liste unterstützt damit nun sogenanntes *Lazy Loading* und ist auch bei sehr großen Ereignismengen ohne spürbare Einschränkungen in der Performance bedienbar. Über weitere Plugins aus dem Backgrid-Framework konnten zudem zusätzliche Filterfunktionen zur Einschränkung aufgrund einer gewünschten Gruppe, eines gewünschten Sensors oder auch einer frei wählbaren Ereignisklassifikation implementiert werden. Die Tabelle kann zudem wieder nach beliebigen Regeln sortiert werden. Auch die Detailansicht einzelner Ereignisse erfuhr eine derzeit insbesondere graphische Überarbeitung, was in einer besseren Lesbarkeit der präsentierten Daten und einer besseren Integration in das allgemeine Anwendungskonzept resultierte.

Da die genannten Arbeiten zum Zeitpunkt dieses Berichtes noch nicht ganz abgeschlossen sind, hat eine Verteilung der Neuerungen an die Betreiber der Testumgebungen noch nicht stattgefunden. Mit dem

Umbau des Ereignis-Handlings wurde auch eine Überarbeitung des Dashboards nötig, das der Nutzer als Übersicht nach dem Login zu Gesicht bekommt. Die dort gelisteten Statistiken basierten ebenfalls auf der Präsenz aller Ereignisse zugleich im Backend der Webanwendung. Das ist mit Nutzung der lazy-loading-basierten PageableCollection allerdings nicht mehr der Fall, weswegen eine zusätzliche Schnittstelle zum Abrufen von Statistiken notwendig wird. Deren Implementierung ist für den Beginn der kommenden Projektphase geplant.

Testbetrieb Der ausführliche Testbetrieb begann im Februar innerhalb des Sächsischen Verwaltungsnetzwerkes. Nachdem der dafür nötige Server bereits im Vorfeld organisiert und ein Mitarbeiter für die Betreuung der Software geschult wurde, begann anschließend die Beschaffung und Verteilung von Sensoren im Netzwerk der Landesverwaltung. Sowohl am Standort des SID in Dresden, als auch in Verwaltungsnetzen in Kamenz und Lichtenwalde, kommen Sensoren für die Testphase zum Einsatz. Deren Installation verlief nach anfänglichen (zuvor beschriebenen) Problemen mit den vorhandenen Proxy-Servern problemlos. Bisher aufgezeichnete Ereignisse bestätigten die Funktionsfähigkeit des Systems und wiesen zunächst auf ungewöhnlich konfigurierte Netzwerkgeräte hin. Kritische Ereignisse blieben bisher allerdings aus.

Über den Verlauf des Testbetriebes innerhalb des Campusnetzes der TU Dresden informierte im April bereits ein gesonderter Kurzbericht:

Kurzbericht über den HoneySens-Einsatz an der TU Dresden (April 2016)

Das HoneySens-Sensorsystem zum Auffinden von Bedrohungen im Inneren eines IT-Netzwerkes wurde in Vorbereitung eines umfangreichen Testlaufs an der TU Dresden Anfang Februar 2016 in Betrieb genommen. Sensoren wurden zunächst im Netzwerk der Stabsstelle für Informationssicherheit, am Lehrstuhl für Datenschutz und Datensicherheit sowie temporär innerhalb eines für Gäste der Universität reservierten Teilnetzes platziert. Nach einem Einsatzzeitraum von zwei Monaten sind folgende Beobachtungen getroffen wurden:

- Installation und Administration des Systems verliefen im TU-Netz weitestgehend reibungslos und erfüllen somit die Anforderungen an das Projekt. Es wurden ebenfalls verschiedene Anregungen und Verbesserungsvorschläge zur Erweiterung der webbasierten Verwaltungsoberfläche geäußert und in Teilen bereits umgesetzt.
- Es wurden im angegebenen Zeitraum, davon überwiegend innerhalb des Gastnetzes, 64 potentiell sicherheitsrelevante Ereignisse registriert. Darunter befanden sich Verbindungsversuche zu diversen typischerweise ungenutzten TCP- Ports (u.a. 777, 62078, 2987), UDP-Pakete ohne Payload (u.a. Ports 7, 137 und 161), aber auch Pakete, die eindeutig bestimmten Protokollen zugeordnet werden können (darunter HTTP auf unüblichen Ports). Ein Teil der Ereignisse kann leicht als Scanversuch interpretiert werden, da binnen kurzer Zeit verschiedene Protokolle auf unterschiedlichen Ports abgefragt wurden. Einige dieser Vorfälle könnten zudem Indikatoren für Angreifer innerhalb des Netzes sein (UDP/1900, TCP/445 und UDP/2054 mit binären Nutzdaten). Eine intensivere Untersuchung dieser Vorfälle war aus Datenschutzgründen im Rahmen des Forschungsprojektes nicht möglich.
- Die Mitarbeiter der Stabsstelle Informationssicherheit der TU Dresden schätzen den zusätzlichen Einblick in ihre IT-Infrastruktur und betrachten die gewonnenen Informationen als wertvoll für ihre Arbeit.

Im Laufe des Monats April ist eine Einbindung von zwölf weiteren Infrastruktureinheiten der Universität mit jeweils 2-3 Sensoren geplant. Das Projekt stieß dabei auf viel Zuspruch bei den für die Netze zuständigen Administratoren.

3 Ausblick

In den kommenden Wochen genießt die Ausweitung des Testbetriebs beider Teststellungen höchste Priorität. Seitens der Weiterentwicklung steht zudem die Fertigstellung der neuen Ereignisübersicht und des Dashboards im Fokus. Fehlermeldungen und Anregungen von den das System nutzenden Administratoren werden zudem weiterhin berücksichtigt und, entsprechend priorisiert, in das System eingearbeitet. Im Rahmen des sich ausweitenden Testbetriebes wurde vermehrt nach einer bebilderten Anleitung oder Dokumentation der Software aus Nutzersicht gefragt. Die Erstellung einer solchen als zusätzliches Handbuch für die Administratoren ist somit ebenfalls geplant, wobei längerfristig die Integration der Dokumentation in einer interaktiven Form direkt in die Webanwendung in Erwägung zu ziehen ist. Eine kurze Anleitung für Installation und Aktualisierung der Serversoftware ist bereits vorhanden, wird aber gemäß dem Entwicklungsfortschritt noch um zusätzliche Abschnitte erweitert werden.

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
333	Server-side implementation of gateway and DNS support for sensors
334	- Sensor-side implementation of gateway and DNS support - Sensor software version raised to 0.1.3 for all components (including the platform-specific ones). Those should stay in sync in the future. - Web UI: Gateway and DNS fields are now optional
335	- Server version bumped to 0.1.2 - Apache config updated with a redirect to HTTPS (for HTTP) and SSL settings from bettercrypto.org
336	The sensor gets now passed the entire certificate chain for the TLS server certificate. Previously it just received the server cert itself and couldn't verify in case it wasn't self-signed.
337	- Server version bump to 0.1.3 - Added proxy authentication settings to sensor config file (currently unused) - Sensor version bump to 0.1.4 - Polling script bug fixed: htpdate now properly uses proxies if required to - Made recon mode more stable when processing packets without payload - Fixed a bug that prevented the correct application of enabling and disabling proxy settings
338	- Dependency cert-chain-resolver.sh removed, because it doesn't work in an isolated environment without internet access. Apache supports certificate data including intermediate certs for the SSLCertificateFile directive from version 2.4.8 on. Users are now expected to overwrite /etc/ssl/certs/ssl-cert-snakeoil.pem with a file containing the full certificate chain using docker volumes when creating a new container. The sensor config creation worker will then use that file as cert bundle that is distributed to sensors. - Recon service now respects proxy settings by ignoring data coming from the configured proxy (leads to false positives otherwise).
340	- Server updated to 0.1.4.rc1 - Docker build process changed to install a newer apache version from Ubuntu vivid, because we rely on features from apache >=2.4.8 - The docker image now relies on phuison/baseimage:0.9.18
341	- Proxy support for the bbb update script - Server is now more liberal on timestamp checks on sensor polling attempts: differences of up to two minutes are accepted (helps with inprecise clocks) - Server version raised to 0.1.4.rc2
342	The recon service now ignores packages from the system-wide configured DNS servers

343	Gracefully handle network reconfiguration when applying a sensor config (apply_config.py).
344	HTTP(S) basic proxy authentication support for sensors
345	Server-side implementation of HTTP(S) proxy basic authentication support (model + GUI)
346	Proxy support for the sensor configuration creation worker
347	Still server v0.1.4: event filters, users and divisions do now „smart“ updates during regular polling to avoid that models are invalidated while some user is currently modifying them.
348	Still server v0.1.4: DB schema update functionality (as GUI option, supposed to be ran after server updates)
349	Still server v0.1.4: Fixed a bug that led to sensor status data with the correct timestamp being rejected
350	Sensor version bump to 0.1.5
351	- python-requests replaced by pycurl for flexible proxy authentication support (including NTLM) - Dependency on htpdate removed, HTTPS time sync is now handled by poll.py - External python dependencies moved to 'vendor' directory - Common utils.py added for commonly used functions - HTTPS requests being a part of that
352	- Server version bump to 0.1.5 - The server now returns the proxy password to sensors (exclusively) - Sensor polling script refreshes proxy authentication data (still part of sensor sw 0.1.6)
353	Still firmware 0.1.6: proxy authentication forced to NTLM, because curl's ANYAUTH didn't work under all circumstances. This is to be replaced later with some sort of brute-force auth mechanism that just tries different auth schemes until one works.
354	Sensor firmware 0.1.7: Added proxy auth scheme brute-force method that kicks in if the curl autodiscovery didn't work. This helps with some niche proxy configurations. The last successful proxy auth scheme is now also cached in the local file „cache.cfg“, so that we don't have to do the discovery algorithm for every single request.
355	Still firmware 0.1.7: Brute-force is only done if proxy auth credentials were provided
356	Transitioning from DataTables to Backgrid: Event list is now backed by Backgrid, but doesn't support pagination or filtering yet. DataTable code is kept for reference.
357	Backgrid.js updated to 0.3.7 from github, including some manual fixes from https://github.com/wyuenho/backgrid/pull/618 to fix asc/desc table headers

358	Pagination and sorting (server-side) added to the event list. The API now supports additional sort criteria: sort_by, order, page and per_page. It also returns the number of total (accessible) events for each request.
359	- Event list filtering per group, sensor and classification added - Event details migrated from modal dialog to sliding overlay - The JS event model now doesn't overwrite its raw timestamp value with an object, because this isn't correctly supported by the lazy-loading collection (initialize() isn't called) - Paginator styling now does match Bootstrap themes better