Whitepaper

Sicherheit enaio mobile, enaio webclient und enaio desktop webclient

Version 3.1: Juni 2020
Autor: René Espenhahn

Einleitung

Mobile Anwendungen werden überwiegend unterwegs verwendet. Sie laufen auf kleinen, portablen Smartphones oder Tablets. Aufgrund der Größe und Portabilität sind sie jedoch leicht durch Dritte in einem unaufmerksamen Moment zu entwenden. Eine Entwendung hat oftmals nicht nur die Folge, dass das mobile Endgerät verloren ist, sondern auch die auf dem Gerät gespeicherten Daten. Der Wert sensibler persönlicher Daten oder Unternehmensdaten ist oftmals beträchtlich größer als der Wert des eigentlichen mobilen Endgeräts. Aus diesem Grund müssen mobile Anwendungen, die mit sensiblen Daten arbeiten, auch bei einem Diebstahl des mobilen Endgeräts sicherstellen, dass diese nicht in die Hände Dritter fallen.

Neben dem materiellen Diebstahl des mobilen Endgeräts existiert auch der virtuelle Diebstahl von Daten durch bösartige Apps, die aus Unkenntnis oder durch Dritte auf dem mobilen Endgerät installiert wurden. Solche Apps lesen oft sensible Daten, die auf dem mobilen Endgerät verarbeitet werden, aus und senden sie an den Hersteller der bösartigen App, der diese für seine Interessen nutzt.

Alles was für mobile Apps gilt, gilt in gleicher Weise auch für Webanwendungen wie dem enaio webclient. Auch dieser kann mobil auf dem Smartphone, Tablet oder Laptop einfach im Browser genutzt werden und es gelten somit dieselben Sicherheitsansprüche wie für die mobilen Apps.

Durch clientseitiges Skripting kann der enaio webclient und enaio mobile stark angepasst werden. Clientskripte werden dazu auf dem Rechner oder mobilen Endgerät des Anwenders ausgeführt. Dies zieht mehrere sicherheitsrelevante Aspekte mit sich wie die Übertragung der Skripte und die sichere Ausführung dieser, sodass die Ausführung seitens des Anwenders oder eines Dritten nicht manipuliert werden kann. In diesem Whitepaper wird ebenfalls darauf eingegangen, wie diese Sicherheit gewährleistet wird.

Sicherheitskonzept von Apple iOS

Im Betriebssystem iOS von Apple werden alle Apps in separaten Sandboxes installiert und ausgeführt. Das Sandboxkonzept schirmt die Apps auf dem Betriebssystem voneinander ab. Einer App ist es somit nicht möglich auf Daten anderer Apps zuzugreifen. Dieses Sicherheitskonzept seitens des Betriebssystems beugt bereits dem virtuellen Datendiebstahl von Apps vor, ohne dass in den installierten Apps spezielle Vorkehrungen getroffen werden müssen.

Zusätzlich dazu bietet iOS die Einrichtung eines Pincodes oder einer biometrischen Authentifizierung zur Entsperrung des Betriebssystems an. Ist eine solche Authentifizierung eingerichtet kann das Endgerät ohne diese Authentifizierung nicht verwendet werden. Weiterhin bietet das Betriebssystem eine Option an, dass alle Daten auf dem mobilen Endgerät gelöscht werden, wenn die Authentifizierung zehnmal falsch eingegeben wurde. Diese Sicherheitsfunktionen schützen bereits rudimentär vor der Entwendung sensibler Daten durch Dritte im Falle eines Diebstahls des mobilen Endgeräts. Sie setzen jedoch voraus, dass der Anwender die Sicherheitsfunktionen aktiviert hat und das etwaige Passwort einen hohen Komplexitätsgrad hat.

Viele Daten werden über Funknetze auf das mobile Endgerät übertragen. Hier unterstützt das Betriebssystem die gängigen Verschlüsselungsverfahren für Funktechnologien WEP, WPA (Enterprise), WPA2 (Enterprise) sowie VPN. Hierdurch können die Datenverbindungen zwischen dem mobilen Endgerät und den Accesspoints gesichert erfolgen. Bei einer Verbindung über das Mobilfunknetz werden die Daten mittels A5/3 verschlüsselt übertragen.

Sicherheitskonzept von Google Android

Identisch zu iOS verwendet Android ebenfalls ein Sandboxkonzept für jede App, sodass diese mit ihren Daten voneinander abgeschirmt sind. Ebenfalls kann auch unter Android ein Pincode oder eine biometrische Authentifizierung zur Entsperrung des mobilen Endgerätes eingerichtet werden. Für den grundlegenden Schutz der Daten sorgt somit das Betriebssystem ähnlich wie bei iOS. Auch die Verschlüsselung der Datenübertragung ist identisch zu iOS.

Sicherheitskonzept von enaio mobile

Zusätzlich zum betriebssystemseitigen Sicherheitskonzept verfügt die App enaio mobile von OPTIMAL SYSTEMS über weitere Sicherheitsfunktionen.

Um Daten über die App vom enaio Server abrufen zu können, muss sich die App an dem enaio gateway anmelden. Die Authentifizierung erfolgt hierbei über Basic Authentication (RFC 2617), die den Benutzernamen und das Passwort als Base64-kodierte Zeichenketten an den enaio gateway sendet.

Da bei dieser Art der Authentifizierung der Benutzername und das Passwort unverschlüsselt über das HTTP-Protokoll versendet werden, sollten sie enaio mobile mittels dem gesicherten HTTPS-Protokoll an das enaio gateway anbinden. Wenn das HTTPS-Protokoll für die Übertragung eingesetzt wird, so erfolgt die gesamte Datenkommunikation zwischen enaio mobile und dem enaio gateway SSL/TLS-verschlüsselt. Der Einsatz von SSL oder seinem Nachfolger TLS wird dringend

empfohlen, da die Verschlüsselung der Datenverbindung im Mobilfunk nur bis zum jeweiligen Accesspoint erfolgt und danach die Daten unverschlüsselt weitergeleitet werden. SSL und TLS gewährleistet eine Ende-zu-Ende Verschlüsselung und Infrastrukturelemente in dieser Verbindung können die Daten nur sehr schwer entschlüsseln.

Die Zugangsdaten zur Authentifizierung des Anwenders am enaio gateway werden innerhalb von enaio mobile mittels des Cordova Plugins *cordova-plugin-secure-storage* sicher gespeichert. Unter iOS werden die Passwörter dazu in der Schlüsselbundverwaltung gespeichert. Android hingegen besitzt betriebssystemseitig kein Äquivalent zur Schlüsselbundverwaltung. Aus diesem Grund werden die Passwörter hier mittels einer AES Verschlüsselung gespeichert.

Ist das mobile Endgerät während der Anmeldung offline, so werden die eingegebenen Anmeldedaten gegen die zuletzt für das ausgewählte Profil hinterlegten Anmeldedaten validiert. Stimmen diese überein, so wird enaio mobile offline aus dem Cache gestartet.

Bei erfolgreicher Authentifizierung beim Start von enaio mobile am enaio gateway werden von der laufenden App im Onlinefall viele empfangenen Daten in einem Cache zwischengespeichert. Die Daten werden ebenfalls innerhalb der Sandbox der App abgelegt und sind durch die Sandbox vor dem Zugriff Dritter geschützt. Hierbei werden innerhalb von enaio mobile mehrere IndexedDB`s verwendet. Die Cachedaten innerhalb der IndexedDB`s überdauern auch App Neustarts und sorgen z.B. für einen schnelleren Boot ab dem zweiten Mal da nicht alle Daten neu übertragen werden müssen, was ebenfalls Bandbreite spart. Die Authentifizierung beim Start von enaio mobile beugt ebenfalls einem unautorisierten Zugriff Dritter, die das Endgerät entsperrt in die Hand bekommen, auf enaio mobile vor. Es kann zwar ein Autologin eingerichtet werden, aber dies ist für mobile Endgeräte nicht empfohlen.

Neben dem Caching werden Favoriten- und Offlinedokumente offline samt Indexdaten zugänglich gemacht (enaio 9.00 webclient SP2). Diese Funktion ermöglicht es, Dokumente und Indexdaten auch ohne bestehende Internetverbindung bei einem Kundentermin verfügbar zu haben. Wenn ein ECM Ordner oder Register den Favoriten hinzugefügt oder für die Offlinenutzung gekennzeichnet wird, so wird der Ordner oder das Register samt seiner Kindobjekte zur Offlinenutzung synchronisiert. Offline verfügbare Objekte werden ebenfalls in einer IndexedDB, die durch die Sandbox des Betriebssystems vor dem Zugriff anderer Apps gesichert ist, gespeichert. Die Synchronisation und das Speichern der Offlineobjekten erfolgt automatisch, sobald ein Dokument innerhalb der App als Favorit oder zur Offlinenutzung gekennzeichnet wird. Wird ein Objekt mittels enaio mobile aus den Favoriten entfernt oder sein Offlinestatus entfernt, so entfernt enaio mobile das Dokument direkt aus der IndexedDB. Wird ein Objekt über den enaio Client zu den Favoriten hinzugefügt oder aus den Favoriten entfernt, so fügt/entfernt enaio mobile das Objekt bei der nächsten Synchronisierung aus dem Offline Cache.

Sicherheitskonzept enaio desktop webclient

Dem enaio desktop Webclient liegt die Open Source Plattform Electron zu Grunde. Electron wird vornehmlich von GitHub Inc. entwickelt. Identisch zu enaio mobile muss sich ein Anwender mit einem Profil am desktop Webclient anmelden. Die Validierung erfolgt hierbei identisch zu enaio mobile.

Im Gegensatz zu enaio mobile läuft der enaio desktop webclient nicht in einer Sandbox und ist somit unter Windows mit seinen Daten nicht separiert von allen anderen Apps und Programmen. Andere Programme können somit auf alle Daten des enaio desktop webclient zugreifen, genauso wie es der enaio desktop webclient auch bei ihnen könnte. Identisch zu enaio mobile speichert der enaio desktop webclient alle Cachedaten in IndexedDB`s. Das Datenverzeichnis des enaio desktop webclient liegt im Roamingordner des am Betriebssystem angemeldeten Benutzers. Dort finden sich ebenfalls die Datendateien zu den IndexedDB`s. Wenn diese Dateien jedoch näher betrachtet werden, so sind diese, wie es bei fast allen Datenbanken der Fall ist, in einem proprietären Format gespeichert um gute Lese- und Schreibgeschwindigkeit zu bieten. Recherchen unsererseits haben ergeben, dass es keine einfachen Programme gibt, um diese IndexedDB`s zu lesen, und somit sind die Daten innerhalb dieser Datenbankdateien hinreichend sicher. Nichts desto trotz wird dieser Sachverhalt weiterhin untersucht und sollten sich daran Zweifel ergeben, werden weitere Maßnahmen zur Datensicherheit des enaio desktop webclient Cachedaten umgesetzt.

Identisch zu Apple iOS werden Profilpasswörter beim enaio desktop webclient im systemeigenen sicheren Schlüsselbund gespeichert. Enaio desktop webclient nutzt dazu die Open Source Bibliothek Keytar [KT].

Sicherheit von Clientskripten

In den nachfolgenden Kapiteln widmet sich dieses Whitepaper der Thematik der Clientskripte. Es wird gezeigt, wie die Ausführung abgesichert werden kann und wieso dies notwendig ist. Zum Anfang dieser Kapitelserie soll jedoch auf einen zentralen Punkt in Bezug auf Clientskripte hingewiesen werden.

Die nachfolgend dargelegten Konzepte versuchen die Ausführung von Clientskripten unter enaio mobile und enaio desktop webclient sicher zu machen. Es gibt jedoch keine hundertprozentige Sicherheit in der heutigen Welt. Alle Sicherheitsvorkehrungen, die irgendwo getroffen werden, sind ausschließlich dazu da, unbefugten Personen den Zugriff auf die Clientskripte so schwer wie möglich zu machen. Werden sicherheitstechnische Hürden auf einem Computer trotzdem geknackt, so sollten im Hintergrund, auf der Serverseite, weitere Absicherungen existieren.

Alle Anfragen (Suche, Aktualisierung, Neuanlage, Löschen, etc.), wo mittels Clientskripten Felder unsichtbar, schreibgeschützt oder sonstige Sicherheit durchgesetzt werden soll, sollten serverseitig nochmalig quergeprüft werden. Dies hat nicht nur als Hintergrund, dass die Clientskripte kompromittiert werden können. Vielmehr bezieht enaio mobile, enaio webclient und enaio desktop webclient neben vielen anderen Komponenten seine Daten aus öffentlichen Schnittstellen wie dem enaio appconnector. Sämtliche Anfragen, die einer der Clients an den enaio appconnector sendet, können genauso gut manuell durch eine unbefugte Person im Kontext des aktuell angemeldeten Anwenders über diesen abgesetzt werden.

Neben dem enaio appconnector existieren bei enaio viele weitere öffentlich dokumentierte Schnittstellen, über welche Daten am enaio server manipuliert werden können. Dazu zählen z.B. die COM-API, enaio webservices und die Microservicelandschaft mit ihren vielen Services. Diese öffentlichen Schnittstellen existieren heutzutage auch bei einer RichClient-Installation, da sie von dieser ebenfalls benötigt werden. Durch seine proprietäre Art bietet der RichClient einen guten Schutz vor der Manipulation der Clientskripte. Eine unbefugte Person kann jedoch auf die

öffentlichen Schnittstellen ausweichen, um seine Datenmanipulation vorzunehmen. Es ist somit auch hier zwingend erforderlich, dass zur Durchsetzung der Sicherheit alleinig der enaio Server mit seinem Rechtesystem und seinen Serverskripten verantwortlich sein sollte. Alleinige clientseitige Sicherheit bietet ausschließlich trügerische Sicherheit und keine reale Sicherheit.

Sicherheitskonzept für Clientskripte unternehmensintern

Clientskripte bieten die Möglichkeit viele Aspekte von enaio mobile und enaio webclient an die projektspezifischen Anforderungen anzupassen. Andererseits können sie auch zu einer großen Sicherheitslücke werden, wenn deren Ausführung nicht ordentlich gesichert oder sie falsch konzipiert worden sind.

Clientskripte werden, wie ihr Name vermuten lässt, im Browser des Anwenders ausgeführt. Dies bedeutet an jedem Arbeitsplatz eines enaio mobile oder enaio webclient Anwenders. Die Javaskripte werden vom enaio appconnector als JSON abgerufen und dann im jeweiligen Browser und in enaio mobile zur festgelegten Aktion zur Ausführung gebracht. Jeder Browser verfügt zum Testen und Entwickeln von Javaskript eine Entwicklerkonsole, die sich über die Taste F12 öffnen lässt. Über die Entwicklerkonsole eines Browsers lassen sich die Clientskripte im Klartext einsehen und auch temporär modifizieren. Weitergehend kann darüber an einem gesetzten Breakpoint neuer Javaskriptcode mit sofortiger Wirkung ausgeführt und sämtliche HTTP-Aufrufe eingesehen sowie modifiziert zur erneuten Ausführung gebracht werden. Dies sind nur drei Aspekte der Entwicklerkonsole, die sehr viele weitere Möglichkeiten der Manipulation von Webanwendungen bereitstellt.

Während die Entwicklerkonsole im Zuge der Entwicklungs- und Testphase eines Projektes als unerlässliche Hilfe benötigt wird, kann sie in der Produktivphase als Backdoor angesehen werden. Es wird deswegen empfohlen, die Entwicklerkonsole über Gruppenrichtlinien unter Windows oder Konfigurationsdateien Betriebssystemübergreifend unzugänglich für den Großteil der Anwender zu machen. Die von enaio mobile und enaio webclient unterstützten Browser bieten hierzu alle ein bis zwei konfigurierbare Möglichkeiten an.

Eine durchgängige Möglichkeit, die jedoch auf Microsoft Windows Betriebssysteme beschränkt ist, ist die der Gruppenrichtlinien. Jeder Browserhersteller bietet einen eigenen Satz an Gruppenrichtlinien an, über welche sich die Entwicklerkonsole deaktivieren lassen. Neben den Gruppenrichtlinien für Windows stellen Mozilla und Google für ihre Browser auch die Möglichkeit der Deaktivierung über eine Konfigurationsdatei an, die an einem Ort im Dateisystem liegt, auf den der Anwender keine Schreibrechte hat, um sie zu ändern.

Im Mozilla Firefox führt Aktivierung der Gruppenrichtlinie [MOZ] *Werkzeuge für Webentwickler deaktivieren* dazu, dass alle Anwender dieser Gruppenrichtlinie die Entwicklerkonsole nicht mehr öffnen können. Alternativ kann im Unterverzeichnis *distribution* vom Firefox eine Konfigurationsdatei namens *policies.json* mit folgendem JSON Inhalt hinterlegt werden:

```
{
        "policies": {
                "DisableDeveloperTools": true
        }
}
```

In den Gruppenrichtlinien für den Google Chrome [GC1] muss *Festlegen, wo Entwicklertools verwendet werden können* aktiviert und mit der Option *Nutzung der Entwicklertools nicht zulassen* versehen werden. Unter Linux oder Mac OS X kann ein Konfigurationsschlüssel [GC2] gesetzt werden, um die Entwicklerkonsole im Chrome Browser unzugänglich zu machen.

Der Microsoft Edge Browser verfügt ausschließlich über die Möglichkeit der Sperrung via Gruppenrichtlinie. Die betreffende Richtlinie wird bereits von Windows mitgebracht und es ist nicht notwendig zusätzliche Richtlinienvorlagen aus einer externen Quelle zu importieren. Beim Edge Browser muss die Gruppenrichtlinie *Windows Components\Microsoft Edge\Allow Developer Tools* deaktiviert werden.

Für den Safari Browser unter Mac OS X konnte im Zuge der Aktualisierung dieses Whitepapers keine Möglichkeit der Deaktivierung gefunden werden. Wir haben bei Apple angefragt und warten hier auf eine Rückmeldung. Mit der nächsten Aktualisierung hoffen wir hier ebenfalls eine Möglichkeit darlegen zu können.

Sicherheitskonzept für Clientskripte Mobile

Unter Apple iOS und Google Android selber existiert auf einem Tablet oder Smartphone keine Entwicklerkonsole, die es gilt zu deaktivieren, um die Sicherheit der Clientskripte zu gewährleisten. Wurde enaio mobile aus dem Apple Appstore oder Google Playstore auf dem mobilen Endgerät installiert, so ist auch kein Remote Debugging über Safari unter Mac OS X oder Google Chrome möglich, da die App nicht als Debuggingziel zur Verfügung steht. Somit sind Clientskripte unter enaio mobile sicher, wenn sie gesichert via HTTPS zu enaio mobile übertragen wurden.

Sicherheitskonzept für Clientskripte externe Mitarbeiter

Arbeiten von Unterwegs und zu Hause ist in der heutigen Arbeitswelt mittlerweile sehr weit verbreitet. Auf das Unternehmensnetzwerk kann somit oftmals auch aus dem Internet zugegriffen werden und darüber hinaus, da enaio webclient eine Webanwendung ist, auch oftmals mit einem Webbrowser auf dem heimischen Computer. Wenn dies in einem Projekt der Fall sein sollte, so sind weitere Sicherungsmaßnahmen für Clientskripte erforderlich, die nun in zwei Szenarien beleuchtet werden.

Das erste Szenario ist, dass von Extern und dem Internet ausschließlich mit mobilen Apps zugegriffen werden darf. In diesem Fall kann der enaio webclient Server, welcher die index.html des enaio webclient ausliefert, so konfiguriert werden, dass er diese nicht ausliefert. Diese Datei bildet den Einstieg in den Webclient und wenn sie extern nicht verfügbar ist, kann der enaio webclient nicht aus dem Internet aufgerufen werden. Die Apps von enaio mobile für iOS und Android sowie der enaio desktop webclient benötigen diese Datei nicht, da sie sie selber mitbringen.

Im zweiten Szenario, wo ein Zugriff auf den enaio webclient aus dem Internet gewünscht ist, um Mitarbeitern von Zuhause oder von Unterwegs Zugriff auf die Unternehmensressourcen zu geben, stellt sich die Absicherung schwerer dar, als nur die Einstiegsdatei im enaio webclient Server zu blockieren. In dieser Konfiguration haben alle Anwender, die die Unternehmensfirewall passieren dürfen und sich am enaio gateway anmelden können Zugriff auf den enaio webclient egal von

welchem Computer sie zugreifen. Private Computer sind im Normalfall nicht in die Unternehmensdomain integriert, sodass auch keine Gruppenrichtlinien bei diesen angewendet werden können. Somit greifen die zuvor vorgestellten Möglichkeiten der Deaktivierung der Entwicklerkonsole in den Browsern bei privaten Computern nicht. Es ist auch so nahezu unmöglich private Computer abzusichern, weswegen wir hier nachfolgend zwei Alternativen aufzeigen.

Die erste Alternative ist jene, dass sich alle externen Mitarbeiter auf einen Terminalserver von extern gesichert einwählen müssen und auf diesem dann remote arbeiten. Die Terminalserversitzung ist dann wieder unternehmensintern und die verwendeten Browser können über Gruppenrichtlinien, wie zuvor beschrieben, abgesichert werden.

Alternative zwei ist, dass das Netzwerk zwar zum Internet gesichert geöffnet ist, aber Mitarbeiter sich ausschließlich mit Unternehmenslaptops am enaio gateway anmelden dürfen. Auf den Unternehmenslaptops wird dazu mit Clientzertifikaten gearbeitet, die auf den berechtigten Laptops im Betriebssystem hinterlegt sind. Vor dem enaio gateway wird dann ein Clientzertifikatsproxy eingerichtet, der diese Verbindungen annimmt und validiert. Somit haben private Computer, die Mitarbeiter zu Hause besitzen keinen Zugriff auf den enaio webclient während Unternehmenslaptops, die die Mitarbeiter zum Arbeiten mit nach Hause genommen haben, Zugriff auf den enaio webclient haben. Da dieses Szenario recht komplex werden kann, wenden sie sich zwecks Informationen an unseren Support oder in Projekten an die Abteilung PRDEV von OPTIMAL SYSTEMS.

Der enaio webclient läuft ebenfalls auf Tablets und Smartphones im mobilen Browser. Wie zuvor beschrieben ist enaio mobile ohne weiteres Zutun auf diesen mobilen Endgeräten als sicher zu betrachten. Mobile Browser sind hingegen nicht sicher. Unbefugte Personen können das Tablet oder Smartphone via USB oder WLAN mit einem Computer verbinden, der nicht zum Unternehmensnetzwerk gehört und somit nicht den Gruppenrichtlinien unterliegt. Durch die Aktivierung des Remotedebuggings auf dem mobilen Endgerät und die Verbindung zum Computer kann auf dem Computer remote Debugging erfolgen. Im Apple Safari, Google Chrome oder Mozilla Firefox auf dem Computer kann dann die Entwicklerkonsole für den Browser auf dem mobilen Endgerät geöffnet werden und der enaio webclient, der auf dem mobilen Endgerät im Browser läuft und sich ggf. sogar über ein Clientzertifikat gegenüber beim enaio gateway autorisiert, manipuliert werden.

Dieses Szenario ebenfalls zu abzusichern erfordert eine Kombination mehrerer hier vorgestellter Techniken. Einerseits sollten nur mobile Endgeräte auf den enaio webclient zugreifen dürfen, die z.B. über ein gültiges Clientzertifikat verfügen. Weiterhin muss das Remotedebugging auf dem mobilen Endgerät unterbunden werden. Zur Unterbindung des Remotedebuggings bedarf es eines Mobile Device Managements (MDM). MDMs können Richtlinien auf mobilen Endgeräten durchsetzen wozu auch gehört, dass Remotedebugging dauerhaft deaktiviert sein muss oder soll. Muss bedeutet, dass der/die Browser nicht mehr starten solange diese Betriebssystemeinstellung des Remotedebuggings aktiviert ist. Soll bedeutet, dass die Betriebssystemeinstellung seitens des Benutzers nicht geändert werden kann und abgeschaltet ist. Auf dem Markt existieren mehrere MDM Hersteller, die sie diesbezüglich beraten können. MDMs sind darüber hinaus sehr für die firmeneigene IT zu empfehlen, da sie viele weitere sicherheitsrelevante Richtlinien zur Durchsetzung auf den firmeneigenen mobilen Endgeräten anbieten und darüber hinaus die Softwareverteilung auf den Geräten der Mitarbeiter steuerbar machen.

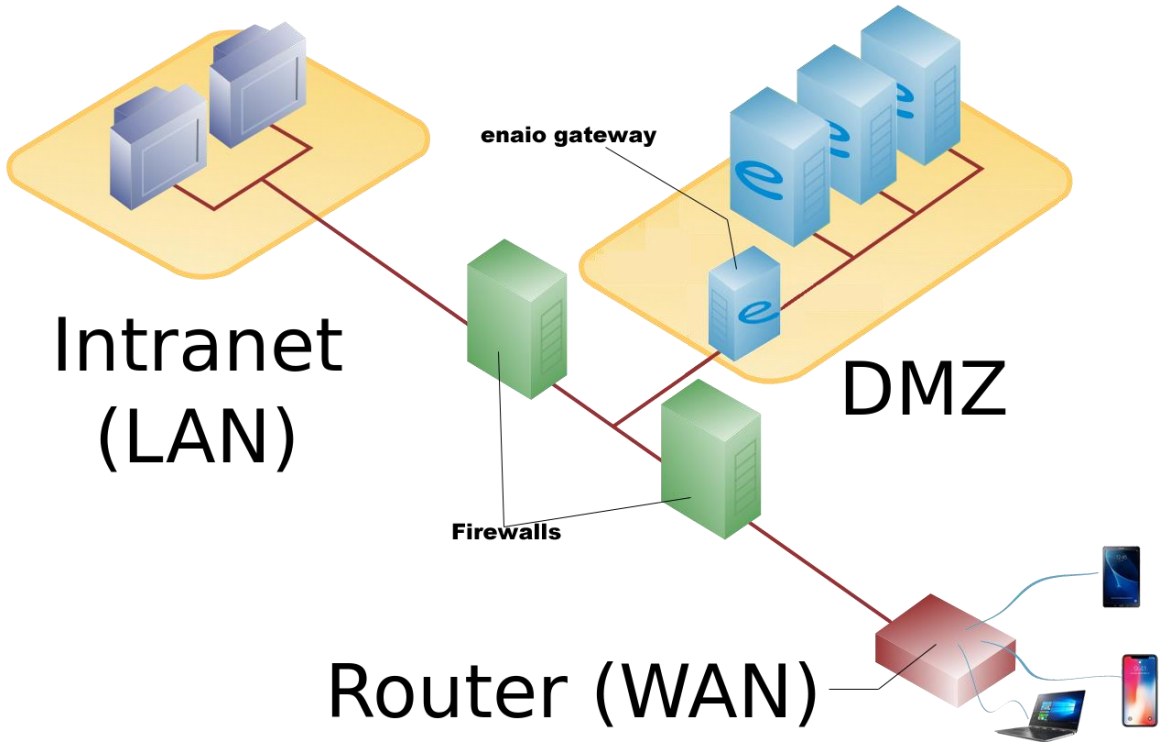
IT-Sicherheit für die enaio services im Unternehmensnetzwerk

Enaio mobile und enaio webclient kommunizieren per HTTP/HTTPS mit dem enaio gateway. Hinter dem enaio gateway benötigt enaio mobile und enaio webclient mehrere Services:

- enaio webclient service
- enaio appconnector service
- enaio documentviewer service
- enaio license microservice
- enaio search microservice
- enaio index microservice

Um die Sicherheit der Daten durch das enaio gateway zu den einzelnen Services oder Microservices zu gewährleisten, sollten System und Netzwerk gemäß den IT-Grundschutzkatalogen [ITG] des Bundesamts für Sicherheit in der Informationstechnik aufgebaut und konfiguriert sein.

Die nachfolgende Grafik veranschaulicht den Zugriff von enaio mobile und enaio webclient auf die genannten services und microservices innerhalb einer Demilitarisierten Zone (DMZ).



Die Grafik auf der vorherigen Seite zeigt sowohl den Zugriff auf die enaio services aus dem internen Unternehmensnetzwerk als auch jene von mobilen Mitarbeitern aus dem Internet. Das BSI empfiehlt ein zweistufiges Firewall-Konzept zum Internet. In diesem Fall trennt eine Firewall das Internet von der DMZ und eine weitere Firewall die DMZ vom internen Netz. Dadurch kompromittiert eine einzelne Schwachstelle noch nicht gleich das interne Netz. Im Idealfall sind die beiden Firewalls von

verschiedenen Herstellern, da ansonsten eine bekannte Schwachstelle ausreichen würde, um beide Firewalls zu überwinden.

In der DMZ stehen dann die enaio Services zur Verfügung. Verbindungen zu diesen gehen in einer enaio mobile und enaio webclient Installation allesamt durch das zentrale enaio gateway. Erst das enaio gateway routet die Anfragen zu den eigentlichen enaio services und enaio microservices. Wird mit Clientzertifikaten gearbeitet, so muss vor dem enaio gateway noch ein Reverseproxyserver installiert werden, der wie das enaio gateway alle Anfragen kanalisiert, das Clientzertifikat auswertet und bei Validität die Verbindung an das enaio gateway weiterleitet.

Wenn das Corporate-Netzwerk aus mehreren verteilten Knoten besteht (z.B. Rechenzentrum Stuttgart, München, Hamburg usw.), sollten alle in der Grafik gezeigten Elemente des Corporate-Netzwerks in jedem Knoten existieren. Der Webserver mit Reverse Proxy würde dann je nach Anfrage-URL die Anfrage an einen speziellen Knoten übersetzen.

Einsatz von Mobile Device Management Lösungen

Enaio mobile nutzt das Cordova Plugin *cordova-plugin-emm-app-config* der AppConfig Community [APC]. Über dieses lassen sich standardisiert Mobile Device Management (MDM) Lösungen einbinden. Als Administrator haben Sie darüber die Möglichkeit einerseits mobile Endgeräte in ihrem Unternehmen in ihrer Verwendung zu beschränken und andererseits auch enaio mobile ein Verbindungsprofil beim Starten mit zu geben.

Sollten Sie enaio mobile in ein MDM Tool einbinden, so zeigt ihnen dieses konfigurative Felder für die Profilverwaltung an. Sie können die Serveradresse und den Benutzernamen sowie das Passwort für ihr enaio System dort festlegen. Woher sie diese Daten für das Profil laden bleibt Ihnen überlassen. Sie können eine statische Quelle nehmen oder den Benutzer identifizieren und dynamisch seine Logindaten enaio mobile übermitteln. Auch können Sie festlegen, dass der Benutzer direkt angemeldet werden soll. In diesem Falle sieht der Benutzer gar keinen Profilmanager und kann enaio mobile direkt produktiv nutzen.

Während der Entwicklung haben wir mit IBM MaasS360 sowie VMWare Airwatch getestet. Es sollten jedoch auch alle anderen MDMs, wo der Hersteller Mitglied in der AppConfig Community ist funktionieren.

Weitere Sicherheitsfunktionen für enaio mobile und enaio webclient

Die App enaio mobile wird von OPTIMAL SYSTEMS ständig weiterentwickelt. Im Zuge der Weiterentwicklung nehmen wir gerne Ihre Wünsche entgegen und prüfen, gerne auch mit Ihnen zusammen, ob und wie diese in kommende Versionen umgesetzt werden sollen. Wenden Sie sich dazu bitte an unser Produktmanagement pm@optimal-systems.de. Sollten Sie Fragen zur Sicherheit haben, so wenden Sie sich bitte an unseren Support unter support@optimal-systems.de.

Verweise

[KT] Projekt Keytar zur sicheren Speicherung der Profildaten unter enaio desktop webclient: https://github.com/atom/node-keytar

[MOZ] Gruppenrichtlinienkonfiguration zum Verbot der Entwicklertools im Mozilla Firefox: https://github.com/mozilla/policy-templates/releases

[GC1] Gruppenrichtlinienkonfiguration zum Verbot der Entwicklertools im Google Chrome: https://www.chromium.org/administrators/policy-templates

[GC2] Konfigurationsschlüssel zum Verbot der Entwicklertools im Google Chrom ohne Gruppenrichtlinien:
http://dev.chromium.org/administrators/policy-list-3#DeveloperToolsAvailability

[MSE] Gruppenrichtlinienkonfiguration zum Verbot der Entwicklertools im Microsoft Edge: https://www.tenforums.com/tutorials/106201-enable-disable-microsoft-edge-developer-tools-windows-10-a.html#option2

[ITG] IT-Grundschutz, Bundesamt für Sicherheit in der Informationstechnik:
https://www.bsi.bund.de/DE/Themen/ITGrundschutz/itgrundschutz_node.html

[APC] AppConfigCommunity
https://www.appconfig.org