Executive Summary

MobileIron Access allows managed mobile devices to securely access protected enterprise computing services that are hosted within the cloud, or on-premises behind a corporate firewall, or both. Access is comprised of the Administrative (Admin) portal that is a cloud service hosted by Amazon Web Services (AWS), and the Gateway that runs as a separate web-based service within the Standalone Sentry, version 8.0.1 and newer. The Sentry normally resides on-premises within the DMZ or within an internal protected network. Sentry can also be hosted within cloud computing service providers like Microsoft Azure or AWS.

The Access Gateway is a SAML proxy that acts as an Identity Provider (IdP) for the Service Provider (SP), and as a Service Provider (SP) for the Identity Provider (IdP). The proxy architecture has the benefit of the user agent app or web browser connecting directly to the Access Gateway before user authentication, or before service login, enabling various security controls based on user, user group, device, and app. Access also leverages the SAML single sign-on (SSO) standard for federated identity services, commonly used by enterprise networks with mobile device deployments.
Product Overview

MobileIron Access is the security policy enforcement point (PEP), that receives the mobile device’s posture assessment information, along with their current compliance status from MobileIron Core, version 9.0.x and newer, or MobileIron Cloud, version R33 and newer.

Within the Core or Cloud Admin portal, enable App Reputation for malware threat protection. Also, deploy MobileIron Tunnel or AppConnect, to enforce additional security policies. MobileIron EMM compliance actions include: send an alert to the user, block email, quarantine the device, or perform a full device wipe. Finally, enable Conditional Access rules within the Access Admin portal, which are used to allow or block access based on user or user group identity; device identity; managed, unmanaged or tunneled app; and network IP addresses or ranges. This guarantees that additional compliance checks are performed by MobileIron Tunnel or AppConnect, and the connection was initiated by an authenticated user, using a trusted app, on a trusted device.

This document describes the security controls implemented by MobileIron Access and its constituents, to protect both data at rest, and data in motion. Before detailing the MobileIron Access security controls, we will start by providing a brief explanation of the SAML concepts, and an overview of all of its components.

SAML Overview

SAML or Security Assertion Markup Language is an Extensible Markup Language (XML) based protocol, that uses security tokens to exchange the authentication and authorization information about the end-user, also called the principal, between the Identity Provider (IdP), and the Service Provider (SP).

The SAML framework is comprised of four components: assertions, protocol, binding, and profiles. Assertions are made up of three types of statements: authentication, attribute, and authorization.

- Authentication assertion response statements sent from the IdP to the SP, are used to validate the principal’s identity.
- Attribute assertion statements contain specific key-value pair (KVP) attribute information about the principal.
- Authorization assertion decision statements identify what the principal is permitted to do on a specific protected resource.

Protocol defines how specific components like assertion statements are packaged for SAML requests and responses that are exchanged between the SAML entities. SAML version 2.0 introduced additional protocols, with the Authentication Request Protocol the most significant.

Binding describes how the SAML protocol request or response statements are transmitted over the underlying protocols. MobileIron Access implements two types of SAML bindings:

- SAML over HTTP bindings are the most commonly used, which includes passive authentication for Office 365. The HTTP Redirect binding is used by the SP to send a query request statement to the IdP. The HTTP POST binding is used by the IdP to transmit the response assertion statement back to the SP.
• SOAP (Simple Object Access Protocol) bindings are used for Office 365 active authentication with Exchange ActiveSync only. SAML statements are encapsulated within a SOAP envelope. The SOAP envelope itself is contained within the Hypertext Transfer Protocol (HTTP) payload.

Profiles simply provide details of how SAML assertions, protocols, and bindings are combined for specific use cases. The Web Browser SSO profile is the most common, and are illustrated in the diagrams that follow.

The Identity Provider or IdP is a web-based service whose job is to validate the identity of the principal, by providing principal authentication to other SPs within a federation. This is facilitated by the IdP service creating, maintaining, and managing the identity information of the principals. The most prevalent IdP used today is Microsoft ADFS (Active Directory Federation Services). Others include Okta, Ping Identity, OneLogin, Shibboleth, Oracle Access Manager, IBM Tivoli Access Manager, SailPoint IdentityNow and IdentityIQ, and Microsoft Azure Active Directory.

The Service Provider or SP provides the principal with a requested protected app, content, or service, after successfully authenticating with the federated IdP. In mobile computing, SPs can host a suite of productivity apps like Microsoft Office 365 and G Suite; secure storage services like Box, Dropbox, Google Drive or OneDrive; information systems like Salesforce; management services like ServiceNow, or on-premises services like SharePoint or SSL VPNs.

MobileIron Core Admin and User portals, as well as, MobileIron Cloud Admin portal are also Service Providers, where SAML can be used to authenticate into the requested resource. The SP is called the Relying Party (RP), and uses the assertions received from the IdP to decide whether or not to allow access to a principal, and what level of authorization to provide on the requested resource.

**SAML SSO Security**

One of the significant benefits of using SAML for mobile devices is the single sign-on (SSO) login standard. SSO enables the ability for principals to securely access multiple protected apps, content and services with a single set of credentials, that is only entered once.

There are two primary methods of implementing SAML SSO: Service Provider Initiated (SP-Initiated) and Identity Provider Initiated (IdP-Initiated). Both of these SAML authentication flows are illustrated below.

• SP-Initiated SSO is the more common use case of the two scenarios and it occurs when the principal attempts to access the SP directly.
• IdP-Initiated SSO is not as common a use case because it requires the principal to already be authenticated with the IdP.

With the benefits of SSO comes potential security risks, especially if only Basic Authentication is used, and the same username and password are reused for multiple apps and services. This can be mitigated by enforcing a more frequent strong password change policy, or implementing X.509 identity certificates, and enabling multi-factor authentication (MFA) for SSO logins.
SAML Web Browser SSO Flow (Without MobileIron Access)

1. The principal attempts to access the protected remote app, content, or cloud service using a web browser.
2. The app or service identifies the principal’s origin (by app subdomain or user IP address), and redirects the web browser back to the IdP’s SSO URL. This is the SAML authentication request.
3. The IdP receives the SAML authentication request, and presents the web browser with a login page. The principal successfully authenticates using their username and password, or X.509 identity certificate, or biometrics, or gestures, or Integrated Windows Authentication, or any combination of the previously mentioned authentication methods.
4. The IdP builds the authentication response in an XML-formatted assertion containing the principal’s username or email address, and invokes its attribute mapper so that attribute names and values used by the SP are the same. The assertion is then encoded by using the IdP’s proof key to sign the response statement. The IdP posts this statement using the web browser to the SP.
5. The SP and IdP have a mutual trust relationship already established. Both SAML entities have a copy of their peer’s X.509 certificate unique fingerprint within their respective trust key stores. The SP retrieves the authentication response from the IdP, and verifies it using the certificate fingerprint. The SP then generates an SSO token, invokes its attribute mapper so that attribute names and values used by the IdP are the same, and then it sets the attributes for the issued SSO token.
6. The principal’s identity is established, and the SP redirects the web browser to the protected URL. The web browser maps the SSO token attributes as HTTP headers, and then the principal is granted access to the requested app, content, or service.

Figure 1-SAML Web Browser SSO Flow
MobileIron Access SP-Initiated SAML Flow

1. The principal attempts to access the protected remote app, content, or cloud service using a web browser.
2. The app or service identifies the principal’s origin (by app subdomain or user IP address), and redirects the web browser back to the MobileIron Access IdP proxy SSO URL. This is the SAML authentication request.
3. The MobileIron Access’ IdP proxy receives the SAML authentication request, re-signs the authentication request with its signing proof key, and forwards the request to the web browser. The web browser is then redirected to the IdP. The IdP presents the web browser with a login page. The principal successfully authenticates using their username and password, or X.509 identity certificate, or biometric, or gestures, or Integrated Windows Authentication, or any combination of the previously mentioned authentication methods.
4. The IdP builds the authentication response in an XML-formatted assertion containing the principal’s username or email address, and invokes its attribute mapper so that attribute names and values used by the SP are the same. The assertion is then encoded by using the IdP’s proof key to sign the response statement. The IdP forwards the assertion to MobileIron Access SP proxy. MobileIron Access SP proxy then re-signs the assertion using its signing proof key, and posts this statement using the web browser to the SP.
5. The SP, IdP and MobileIron Access have mutual trust relationships already established. All SAML entities have a copy of their peer’s X.509 certificate unique signature fingerprint within their respective trust key stores. The SP retrieves the authentication response from the MobileIron Access IdP proxy, and verifies it using the certificate fingerprint. The SP then generates an SSO token,
invokes its attribute mapper so that attribute names and values used by the IdP are the same, and then it sets the attributes for the issued SSO token.

6. The principal’s identity is established, and the SP redirects the web browser to the protected URL. The web browser maps the SSO token attributes as HTTP headers, and then the principal is granted access to the requested app, content, or service.

MobileIron Access IdP-Initiated SAML Flow

1. The principal attempts to access the protected remote app, content, or cloud service using a web browser. The web browser still has a valid SAML security token for the requested resource from a previously authenticated session.

2. The IdP builds the authentication response in an XML-formatted assertion containing the principal’s username or email address, and invokes its attribute mapper so that attribute names and values used by the SP are the same. The assertion is then encoded by using the IdP’s proof key to sign the response statement. The IdP forwards the assertion to MobileIron Access SP proxy. MobileIron Access SP proxy then re-signs the assertion using its signing proof key, and posts this statement using the web browser to the SP.

3. The SP, IdP and MobileIron Access have mutual trust relationships already established. All SAML entities have a copy of their peer’s X.509 certificate unique signature fingerprint within their respective trust key stores. The SP retrieves the authentication response from the MobileIron Access IdP proxy, and verifies it using the certificate fingerprint. The SP then generates an SSO token, invokes its attribute mapper so that attribute names and values used by the IdP are the same, and then it sets the attributes for the issued SSO token.
4. The principal’s identity is established, and the SP redirects the web browser to the protected URL. The web browser maps the SSO token attributes as HTTP headers, and then the principal is granted access to the requested app, content, or service.

SAML Security Considerations

The SAML request from the SP normally does not contain private data. The request itself is usually not persisted for use later, so there is little need to encrypt the statement itself. Signing the statement will allow the receiver to verify that it has not been modified during transit, and transmitting it over a TLS-protected connection will provide the all-important privacy. Encrypting the request is not required.

The SAML response from the IdP is different, depending on what it contains within the statement. A SAML response that contains security tokens or assertions can include private data. Depending on how these assertions are being used in your system, they may be passed around between different security domains. Some may contain the keys to decrypt the contents because they have a trust relationship established with the SAML provider. These SAML assertions can also be stored in cache memory, or within a database file system, so the threat of unauthorized snooping is there. In this scenario, signing and encrypting the SAML assertions response is recommended, along with using a TLS-protected connection to secure the transmission.

*MobileIron Access does not currently support encryption of the SAML responses from the IdP, but by default enables TLS-protected sessions for both inbound and outbound connections, to and from the IdP and SP.*

SAML Metadata Security

The SAML metadata ensures a secure transaction between the IdP and the SP. It can contain the HTTP binding locations, the signing proof and encrypting public keys, organization name, SSO URL, and Single Logout URL.

When an IdP receives an authentication request from the SP, the IdP checks its list of trusted SPs within its metadata before issuing an authentication response. The metadata also contains the SP’s URL endpoint references to redirect the principal to the proper SP resource, that is part of the authentication response from the IdP. Conversely, the SP checks its metadata when it receives an authentication response from a trusted IdP, by validating the signature on the assertion, which is the proof key used by the IdP for signing its assertion statements.

Since the metadata information for the IdP, SP and MobileIron Access are physically uploaded to the Access Admin portal, it is not necessary to sign or encrypt the metadata file since it is never transmitted over the network. Also, there is no private data contained within the metadata files.
MobileIron Access Admin Portal Security

Secure Multi-Tenancy Architect
The MobileIron Access Admin portal service infrastructure implements a multi-tenancy architecture that enables multiple customers to share physical hardware systems based on geography, that implement powerful logical separation and database segmentation to isolate customer data, and keep it secure. A consistent, built-in security layer governs all access to the customer user data. All access, whether directly or through the API, funnels through this single security point at the application server. Each tenant has a unique identifier that is coded into every data, or metadata object that is irrevocably linked to their associated MobileIron Access Admin portal tenant.

Every administrator account has a unique user identification (ID) that is associated to only one tenant. There is no direct access to the database without going through multiple security layers. Even if access were inadvertently granted to unauthorized personnel, they would still need to break through additional security safeguards, specifically the cryptographic systems used for customer content and certain data types at rest in the MobileIron Access Admin portal service.

Third-Party Cloud Computing Service Security
The MobileIron Access Admin portal tenants are hosted globally by Amazon Web Services (AWS). AWS implements robust security practices by placing next-generation firewalls NGFW), and intrusion detection and prevention systems (IDPS) at their network ingress and egress points. AWS also employs a stringent physical access policy into their data centers and colocation sites. It is compliant with industry and government requirements for security, privacy, and confidentiality of data. Some of the IT Security standards they adhere to are SOC 1, SOC 2, SOC 3, FISMA, PCI DSS Level 1, FedRAMP High Authorization, and FIPS 140-2 compliance.

Hosted Data at Rest Security
The MobileIron Access Admin portal service employs today’s strongest cryptography to encrypt specific data types at rest on disk, and within the database file system. At the user level, passwords are salted with a one-way hash function using SHA 256-bits. The service uses the Advanced Encryption Standard (AES) algorithm to generate a tenant-specific symmetric key with a key size of 256-bits using Galois/Counter Mode (GCM) cryptographic block cipher. The symmetric key that is used to encrypt this data at rest is also encrypted using another unique master key, called the Key Encryption Key (KEK), that also uses AES 256-bit encryption. The KEK is securely stored within the database file system of the specific tenant in MobileIron Access Admin portal.

All data in motion transmitted over the network to and from the MobileIron Access Admin portal service is protected using Transport Layer Security (TLS) versions 1.0, 1.1, or 1.2 cipher suites, by default, although the strongest cipher suites are prioritized to be negotiated first during the TLS handshake.
<table>
<thead>
<tr>
<th>Application</th>
<th>Algorithm Protocol Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption of specific data types at rest</td>
<td>AES 256-bit GCM mode</td>
</tr>
<tr>
<td>Secure communications transmission (HTTPS)</td>
<td>TLS version 1.0, 1.1 or 1.2 cipher suites</td>
</tr>
<tr>
<td>Password hashing</td>
<td>SHA 256-bit (8,000 rounds and 8 character salts)</td>
</tr>
<tr>
<td>Random Number Generation</td>
<td>SecureRandom</td>
</tr>
<tr>
<td>X.509 Public Key Infrastructure and Certificate Revocation List</td>
<td>RFC 5280</td>
</tr>
</tbody>
</table>

*Figure 4-Data at Rest Security*

**Creating Access Admin Portal Administrator Key**

When the company tenant administrator initially accesses their MobileIron Access Admin portal instance using the local administrator credentials provided by their sales representative, they will be prompted to change their temporary password, with a stronger long-term password. This is illustrated in step 1 in Figure-5 below.

The newly-created administrator password is then used to generate the administrator key. This is illustrated in step 2 in Figure-5 below. This administrator key is then used to encrypt the SAML signing proof keys stored in the MobileIron Access Admin portal within the AWS Cloud. This is illustrated in step 3 in Figure-5 below.

The administrator password complexity must meet the following requirements:

- Minimum 8 characters
- No spaces
- At least one lowercase letter (a-z)
- At least one uppercase letter (A-Z)
- At least one number (0-9)
- At least one symbol (!@#$%^&*()-)

*MobileIron does not retain or store the administrator key. Only the customer can decrypt their SAML signing proof keys.*
Sentry Access to SAML Keys

During the registration with the MobileIron Access Admin portal, the Standalone Sentry uploads its public key. This is over a Transport Layer Security (TLS) protected connection between the Sentry and the Access Admin portal. This is illustrated in step 1 in Figure-6 below.

When the Sentry is added to an Access profile, the administrator password is entered, which allows use of the administrator key. This is also over a TLS-protected connection between the administrative console and the Access Admin portal. This is illustrated in step 2 in Figure-6 below.

The administrator key is then used to decrypt and access the SAML signing proof keys. This same mechanism allows the secure delivery of the SAML keys and other secrets to the Sentry, over another TLS-protected connection. This is illustrated in step 3 in Figure-6 below.
Securely Distributing SAML Keys to Sentry

The MobileIron Access Admin portal allows the secure upload of its TLS server certificate obtained from a trusted third-party certification authority. The server certificate should contain two server hostnames within the Subject Alternative Name (SAN) field. The first hostname is the FQDN of the Standalone Sentry, while the second hostname is the FQDN assigned to the Access Gateway. The X.509 server certificate file must be in PKCS #12 format, and password encrypted prior to uploading to the Access Admin portal. A wildcard TLS server certificate is also supported for testing purposes, but is not recommended for production deployments.

The Access Admin portal can accommodate SAML X.509 signing certificates with its proof key, associated private key, and full chain of trust certificates, within a password encrypted PKCS #12 file. These can be obtained from a trusted third-party certification authority, or a private enterprise public-key infrastructure (PKI), like Microsoft Active Directory Certificate Services. These certificates can be assigned to a Federated Pair trust configuration between the SP, IdP and MobileIron Access. This is illustrated in step 1 in Figure-7 below.

The Access Admin portal can also generate a self-signed X.509 signing certificate for the Federated Pair that uses RSA 2048-bit key length, and SHA 256-bit hash function algorithms, with a 30-year lifetime. It is highly recommended that self-signed certificates be utilized for non-production environments only.
The SAML signing certificates with their proof keys are then securely distributed from the Access Admin portal to all Standalone Sentries over a TLS-protected connection. The secure session is initiated by the respective Sentry to the Access Admin portal, and syncs with the Sentry at 15-minute intervals. This is illustrated in steps 2, 3 and 4 in Figure 7 above.

*There are no automatic IdP or SP signing certificate renewal or rollover mechanisms, or expiration alerting in the currently released version of MobileIron Access. This process is still maintained by the customer network administrator.*
MobileIron Sentry and Access Gateway Security

The Access Gateway runs as a separate web-service on the same physical appliance or virtual machine of the Standalone Sentry. As mentioned earlier, the Standalone Sentry normally resides in the DMZ network, and is protected by the corporate perimeter firewall. Sentry can also be deployed within Microsoft Azure and Amazon Web Services (AWS), for high availability (HA) and disaster recovery (DR) network deployments.

When a new Sentry is registered with the MobileIron Access Admin portal, the connection is initiated outbound from the Sentry. The initial connection session to register the Sentry with the MobileIron Access Admin portal tenant is also protected by TLS protocol versions 1.0, 1.1 or 1.2 cipher suites over protocol TCP, port 443.

*This may require outbound connection rules added to the perimeter firewall. By default, Sentry syncs with the Access Admin portal every 15 minutes.

**Note:** The Sentry will not listen for any inbound connection requests on TCP ports 443 without an Access profile configuration assigned by the tenant administrator within the MobileIron Access Admin portal. Once an Access profile is assigned to the Sentry, mobile devices can then connect to the Sentry using HTTPS secure sessions protected by TLS versions 1.0, 1.1, or 1.2 cipher suites, over protocol TCP, port 443, using Basic Authentication or X.509 identity certificate authentication methods.

The Sentry protects sensitive data types at rest like the Kerberos keytab file, secrets or other credentials within its secure file system using AES 128-bit encryption. The TLS server keys are stored in PKCS #12 format using Triple DES (3DES) password encryption, within the Java key store.

<table>
<thead>
<tr>
<th>Application</th>
<th>Algorithm Protocol Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption of sensitive data at rest</td>
<td>AES 128-bit CBC</td>
</tr>
<tr>
<td>Secure communications transmission</td>
<td>TLS versions 1.0, 1.1 or 1.2 cipher suites</td>
</tr>
<tr>
<td>(SSH, HTTPS and Tunnel)</td>
<td></td>
</tr>
<tr>
<td>Random Number Generation</td>
<td>FIPS 140-2 Level 1 - SecureRandom</td>
</tr>
<tr>
<td>X.509 Public Key Infrastructure and Certificate</td>
<td>RFC 5280</td>
</tr>
<tr>
<td>Revocation List</td>
<td></td>
</tr>
<tr>
<td>Email attachment</td>
<td>AES 256-bit (Docs@Work)</td>
</tr>
</tbody>
</table>

*Figure 8-MobileIron Sentry Security*
Access SSL Configuration
By default, the Sentry is configured to negotiate the strongest cryptographic cipher suites that are available today that are part of the TLS version 1.2 protocol. Secure Sockets Layer version 3 (SSLv3) cipher suites are disabled by default.

Strict-Mode SSL
A client connecting to a server over a TLS-protected session is required to validate the server certificate chain of trust obtained during the TLS handshake. This is defined in RFC 5246. This provides additional mutual authentication between the client and server.

The RFC states: “The server’s certificate must come first in the chain. Each following certificate must directly certify the one preceding it. Because certificate validation requires that root keys be distributed independently, the self-signed certificate that specifies the root certificate authority may be omitted from the chain, under the assumption that the remote end must already possess it in order to validate it in any case.”

*Sentry enables Strict TLS by default to secure MobileIron Access and Sentry outgoing connections.*

Server Name Indication
Server Name Indication (SNI) is a TLS extension, defined in RFC 4366. It enables TLS-protected connections to virtual servers, in which multiple servers for different network names are hosted at a single network IP address. Also, by default, Microsoft ADFS version 3.0 requires SNI-capable clients to communicate with it. ADFS is the IdP in a server role, and MobileIron Access is the SP in a client-role.

*MobileIron Sentry and Access enables SNI by default.*
Conclusion
MobileIron Access, Sentry and EMM platforms, provide the multiple layered security to protect users, their private data on their managed mobile devices, and the secure access to protected apps, content, and services, that are hosted within the cloud, on-premises, or both. This document details the security controls implemented by this MobileIron solution that provides the simple user experience, and the strong security required for today’s mobile computing.
References
Additional Amazon Web Services compliance and security references can be found at the following links.

https://aws.amazon.com/compliance/
http://aws.amazon.com/security/

Statements in this document concerning future prospects, business outlook, and product availability and plans are forward looking statements that involve a number of uncertainties and risks. Factors that could cause actual events or results to differ materially include: sales productivity; possible disruptive effects of organizational changes; shifts in customer demand; perceptions of MobileIron and its prospects; technological changes; competitive factors; unanticipated delays in scheduled product availability dates; general business conditions; delays and inabilities in negotiating third partner partnerships, and other factors. The information in the document should not be relied upon in making purchasing decisions. The information on any future shown is not a commitment, promise or legal obligation to deliver any material, code or functionality. The development, release and timing of any features or functionality described for our products in this document remains at MobileIron’s sole discretion. Future product will be priced separately. This document does not