Enterprise Security with Endpoint Agents

Kevin Foltz(✉) and William R. Simpson

Institute for Defense Analyses, Alexandria, VA 22311, USA
{kfoltz,rsimpson}@ida.org

Abstract. Enterprise security is complicated by the use of mobile devices. These devices roam outside the protections of the enterprise core network. They operate closer to threats while simultaneously being farther from the enterprise, which makes compromise more likely and response more difficult. This paper describes an approach using software agents installed on endpoint devices to maintain security of these devices and their associated enterprise. These agents monitor local activity, prevent harmful behavior, allow remote management, and report back to the enterprise. The challenge in this environment is the security of the agents and their communication with the enterprise. This work presents an agent architecture that operates within a high-security Enterprise Level Security (ELS) architecture that preserves end-to-end integrity, encryption, and accountability. This architecture uses secure hardware for sensitive key operations and device attestation. Software agents leverage this hardware security to provide services consistent with the ELS framework. Additional agents leverage this baseline security to provide additional features and functions. This enables an enterprise to manage and secure all endpoint device agents and their communications with other enterprise services.

Keywords: Enterprise · Software agent · System design · Confidentiality · Integrity · Application security · Security · End-to-End encryption · Mobile device management · Host based security

1 Introduction

Defense of an enterprise and its information against external attacks has moved from the central network to the edge devices. Network monitoring provides a centralized approach where all communications can be intercepted, recorded, and analyzed for malicious intent and modified as needed. However, this is complicated by current threats and operational practices.

Network monitoring can provide important insights about lower layer resources and communications but, with widespread encrypted hypertext transfer protocol secure (HTTPS) and similar protocols, it does not have access to the higher layer content. Web application firewalls (WAFs) attempt to bridge this gap by decrypting content for the server, analyzing and modifying it for security, and passing the clean content to the server. The WAF may even open files and execute code to determine if certain content presents a danger to the receiver. This approach catches many attacks that network monitoring

J. Filipe et al. (Eds.): ICEIS 2019, LNBIP 378, pp. 1–16, 2020.
https://doi.org/10.1007/978-3-030-40783-4_1

and pattern-based detection miss, but it breaks the end-to-end security model, introduces latency in communications, and does not stop all attacks.

Widespread encrypted HTTPS traffic requires a network scanner to act as a central point of decryption. This can be accomplished by sharing server private keys with network appliances on the wire, but such an approach violates end-to-end security by breaking every secure connection within the enterprise. In addition, these network appliances provide central points of attack that enable access to all traffic and allow an attacker to impersonate any entity within the enterprise. Such a network-based approach has critical security flaws.

Moving the defense to the edge of the network offers several advantages. There is no need to break end-to-end secure connections. There is no central point of attack that can compromise all connections and impersonate any entity. The defense tools can operate at the endpoint to detect malicious behavior as it happens and directly respond instead of trying to predict it before it happens based on network traffic and then trying to respond remotely after the damage is done.

The edge defense model does have some drawbacks. The distributed nature of the defense introduces the challenge of coordination and correlation of data. End-to-end security requires new approaches to decrypt data for analysis. Also, software agents at the endpoints, which are often lightweight applications, must perform secure operations and initiate secure communication channels.

Endpoint agent architecture design has seen some work with varying goals. (Wang et al. 2003) describe an agent architecture that preserves battery life of mobile devices. (Assink 2016) describes the potential benefits of using agents for Internet of Things (IoT) applications. (Berkovits et al. 1998) and (Varadharajan and Foster 2003) examine security of agents that migrate between different hosts. (Liu and Wang 2003) describe a secure agent architecture for sensor networks. (Šimo et al. 2009) describe a secure agent architecture for mobile agents with similar security goals. However, its agents operate with their own software-based private keys, so the agent code itself must be carefully protected.

There is a lot of work in the area of mobile agent computing, where agents move from device to device. However, our interest is in monitoring the device itself using agents, not doing computations that move agents across devices.

This paper presents a method for enabling distributed endpoint-based defense while preserving end-to-end integrity, encryption, and authentication of communications across the enterprise. This agent architecture extends (Foltz and Simpson 2019), and it is part of a larger effort to secure information sharing for the United States Air Force (Foltz and Simpson 2017).

2 Enterprise Level Security Baseline

This work uses as a starting point the Enterprise Level Security (ELS) model. ELS is designed for high assurance information systems subject to constant sophisticated attacks (Trias et al. 2016). It then addresses the challenge of integrating endpoint device agents into such an architecture while adhering to and working with the existing ELS concepts, components, and protocols. This section provides an overview of ELS and the integration challenges for agent-based security.

2.1 ELS Overview

The core of ELS is identity management and access control. The goal is to uniquely identify every entity in the enterprise, human and non-human, and use these identities with strong authentication methods to initiate communication. For interactions where data or services are requested, access is determined by data providers using rules based on attributes stored in an Enterprise Attribute System (EAS).

The data owner or service provider is part of the enterprise and is responsible for setting access rules. This preserves some degree of autonomy for the data owners and supports scalability through its distributed architecture. These rules are compared against attributes collected from across the enterprise to compute which entities have access to which resources. This information is provided on-demand to requesters in the form of a secure access token. Requesters must authenticate to the token server to receive an access token, and this token is time-limited and tied to the requester's identity and the target resource.

Requesters then authenticate to the target resource and provide the token for access. The token is checked for validity using a server handler. This handler code is provided by the enterprise to all entities using access controls, and it parses the token and conducts security checks in a standardized way. A token that is valid and contains the proper identity and access information provides a requester access to data or services at the provider.

2.2 ELS Design and Implementation

ELS starts with high level goals and design philosophies, which are successively refined into specific methods and implementation details for the core security functions (Foltz and Simpson 2016b). Some of the highest level tenets include the following:

- The enemy is present: We cannot assume that any defense or boundary will keep attackers out of our systems. We must assume that any component can be compromised and plan accordingly. This drives the approach of a distributed architecture instead of a centralized approach.
- Simplicity: Complexity is the enemy of security, and the simpler a security function is, the easier it is to implement correctly and securely.
- Extensibility: A point solution may be effective now, but enterprises change over time, and we need to plan for this change in our initial design. Although this may contradict simplicity at times, it actually makes things simpler in the longer term by forcing us to find solutions that match the level of abstraction of the problems instead of over-constraining our solutions.

These tenets guide the development of basic concepts for ELS design. These include a number of items related to identity:

- Authentication is implemented by a verifiable identity claims-based process. This is required to address the tenet that the enemy is present, and serves to distinguish valid entities from attackers.

- The verification of identity is by proof of ownership of the private key associated with an identity claim. This elaborates on the authentication process by requiring cryptographic key operations to verify identity.
- Active entities act on their own behalf. This prevents the many vulnerabilities associated with impersonation and proxies, and provides a baseline for non-repudiation and strong attribution to assist forensics and accountability.

These concepts are further refined into specific requirements. For identity, each entity is issued an X.509 certificate that is tied to a public/private key pair. The distinguished name (DN) in the certificate is used as a unique identifier for each entity in the enterprise. These X.509 certificates are signed by a certificate authority (CA) that is part of a public key infrastructure (PKI). This PKI includes root CAs, issuing CAs, online certificate status protocol (OCSP) responders for validity checks, and certificate revocation lists (CRLs) for offline use. All entities are vetted thoroughly before they are assigned a certificate and key pair. All private keys are stored in hardware to prevent duplication and to provide accountability.

Communication uses transport layer security (TLS) with HTTPS as well as other protocols that integrate TLS, such as secure lightweight directory access protocol (LDAPS). The PKI credentials are used within TLS to provide a secure, authenticated communication channel for entities within the enterprise. The use of TLS is restricted further by security concerns. For example, the zero-round-trip-time handshakes of TLS v1.3 are generally not permitted, and cipher suites are carefully selected for desired security properties.

Access to resources is generally provided by a token from a security token server (STS). This access token is formatted according to the security assertion markup language (SAML) version 2.0. This provides fields for the identity, attribute values, validity time window, target resource, and digital signature. It also provides the option for encryption of tokens, as well as other security options. The SAML standard allows many options, but the tokens allowed in ELS are restricted to access tokens of a particular form, which differs from many non-ELS implementations where SAML tokens are used primarily for single sign-on (SSO). These restrictions prevent attacks such as SAML wrapping that exploit the wide degree of freedom available in the standard. This is consistent with the tenets of simplicity and extensibility by using only what we need while conforming to a broader standard that permits later changes.

The standards mentioned here, including PKI, TLS, and SAML, are not fundamental to the ELS concepts, but they are the currently adopted implementation choices, so integration with ELS must use these particular choices in addition to conforming to the overall architectural goals.

2.3 Agent Security Challenges

The ELS model starts with the premise that security is between authenticated active endpoints. However, in reality endpoints are one of the most vulnerable areas of any information system. As a result, ELS requires strong guarantees that the endpoints have not been compromised. For example, a stolen smart card credential compromises an individual, but such a problem is often quickly reported by the person who lost the credential.

A compromised device can monitor user activity and act as the user surreptitiously over long periods of time with no obvious signs to the user. A systematic approach is required to monitor devices for such compromise and malicious behavior. An agent is placed on the device for this purpose.

In addition, the ELS infrastructure includes other types of agents, such as logging and monitoring agents and endpoint device management agents.

The primary challenges for agents in a secure environment are as follows:

- Establishing secure agent communication with external entities.
- Tying agent communication to its host device.

The first challenge requires that all endpoints use the same ELS methods to communicate whether they are a person, server, or other active entity in the enterprise. The agent, as the initiator of communication with a central server, gateway, or collection system, qualifies as such an active entity. It must be secured at a level comparable to a user with a hardware-based PKI credential or a server with a key pair stored in hardware. This is challenging because agents operate differently than normal users, servers, or other active entities.

The second challenge relates to the separate methods of authenticating endpoint requesters and the devices themselves. Users, for example, can use smart cards, and servers can use hardware security modules (HSMs) to authenticate from different underlying hardware platforms or even virtual machines. However, hardware authentication must be through a different means, as it must be tied to the hardware platform itself. The challenge for an agent is to tie the agent to its digital identity, and then tie its digital identity to a hardware-based device identity.

3 Endpoint Agent Architecture Security Fundamentals

This section establishes the baseline conditions for secure agents. This relies on agents that communicate security information between the device and central servers. The following section builds on this baseline to provide additional agents with additional functionality.

3.1 Endpoint Device Agents

With the move from desktops and laptops to mobile devices like phones and tablets, the edge of the enterprise has changed. Gone are the days where employees log in from an enterprise machine on an enterprise network in an enterprise building. Current users can come from personal mobile devices in public spaces through a commercial cellular network. This motivates our first use case of endpoint device management. These endpoints may be mobile devices or more traditional laptops, desktops, and servers. Agents for device management must have a software component for the agent code, but they must also leverage a hardware key store on the device. Unlike standard ELS authentication, where the keys are tied only to the user or other entity using the device, agent authentication must be tied to the user and the device hardware.

Such an agent need not and should not have any embedded security information for authentication, because such information could be easily duplicated or extracted. The agent is similar to a web browser on a desktop. The browser does not itself authenticate to servers. It provides the means for a user to authenticate and request or provide content. The agent is similar in nature. It relies on existing device keys and certificates to authenticate and communicate securely. The source of the agent keys must be the device hardware rather than a portable or external key store, because such agents speak for the device itself and not for some other entity like a person or server that can migrate from device to device. The agent provides the communication channel for the device hardware to communicate security information.

This introduces some complications. First, the agent is a piece of software that is separate from its hardware-based keys. Hence, any agent, real or malicious, that gains access to the real agent's keys can act as that real agent. There are a number of attacks possible between a software instance and the hardware keys it uses. This is similar to the challenge of securing keys in the cloud, which has a similar key and software separation issue. The agent, and endpoint security in general, must rely on the device to monitor itself, including the software on it, because the agent cannot be trusted by itself.

Secure key storage and use (SKSU) on a device, such as a TPM, has the capability to perform attestations, and such an attestation is required to ensure that the device is running the proper agent and other software. The attestation is a report that lists the state of hardware and software on the device and provides a signature using a key associated with the particular SKSU module on the device. The SKSU hardware module serves as the root of trust for all device-based communications, as indicated in Fig. 1 (Foltz and Simpson 2019). The SKSU must itself be trusted as a starting point. From there, security for the device and its software functionality can be provided using attestation reports.

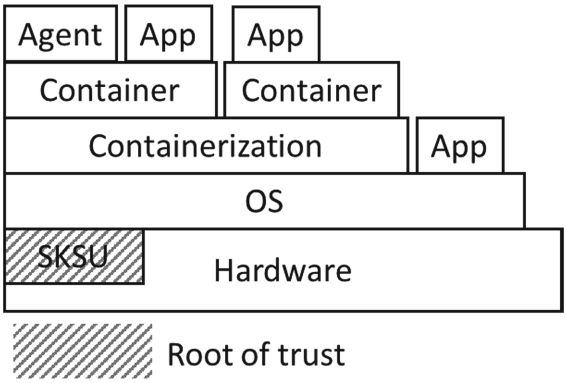

Fig. 1. Using the hardware based SKSU as a root of trust for the device.

The attestation report must cover the hardware, operating system, any virtualization or containerization, and the applications and agents installed on the device. For an agent to communicate securely, it must first produce an attestation report that shows that the device is running as intended at the current time with no malicious entities or

configuration modifications. Typically, this is implemented as a white list of approved software.

The agent invokes the TPM to produce an attestation report with the required parameters. The TPM is an implementation detail that is not important, and it can cause its own problems. The goal is not a separate module that does key operations, but instead an integrated key management capability that is part of the device hardware. For example, a TPM that can easily be removed from a device is not an effective SKSU. Such a hardware element could be placed into a different device that is valid, and a fake hardware element could replace it in the original device. Whenever needed, the TPM output can be captured from the valid device and provided by a compromised device through the fake TPM. This allows the compromised device to look like it is still in a valid configuration. The key problem here is the ability to separate the keys from the device. The keys must be embedded in the device hardware in a way that makes it difficult to extract.

In Fig. 2 (Foltz and Simpson 2019), the elements contained within the attestation report are highlighted. They include the full set of components that can affect the agent, which is running as an app in a container in this case. In this case, the containerization and containers are trusted to isolate the apps within their containers sufficiently well that any other apps or containers are allowed to operate on the device. Other apps outside the container need not be validated, and other containers need not be validated. This might be the case for a phone with separate work and personal spaces. However, if the containerization or containers had known vulnerabilities or insufficient protections and isolation capabilities, then the attestation report would have to include the other components as well. In general, the attestation report must include all elements of the device and its software that could negatively affect the agent's ability to securely communicate with an external entity. This includes modification of the agent as well as attacks that leverage the valid agent's ability to authenticate to external entities.

The trust starts at the bottom with hardware and works its way up the stack. The SKSU validates that the device hardware is operating correctly. It then validates that the operating system is correct. This may include such checks as whether the OS is "rooted," which version is installed, and whether the software is installed properly, such as checking a hash of the executable against a known value. The containerization and applications, including the agent itself, can then be validated in a similar manner.

With a trusted SKSU, and a valid agent running with other valid applications in a valid container in a valid containerization method on a valid operating system on valid hardware, a high degree of trust can be established in the agent functionality. In particular, there is a high degree of trust that a private key operation for the agent was actually initiated by the agent itself. This is required, because there is no external method, such as a PIN or biometric information, to validate the agent's request at the SKSU. The SKSU, in combination with the full validated software stack, is required to secure the private key use by the agent. Without such validation it may be possible for another entity to use the key, which would prevent proper authentication of the agent to the central server.

The agent, with its attestation report, communicates with the external entity, which is often an aggregation point for many device agents. After authentication, the agent

may send a SAML token to the external endpoint for access, in accordance with standard ELS rules for access. A simpler alternative is to have the agent use identity-based authentication. In this case, the server maintains an access control list (ACL) of the known deployed device agents. This reduces the need for a SAML token but eliminates the efficiency that ELS provides for managing access control rules for large groups.

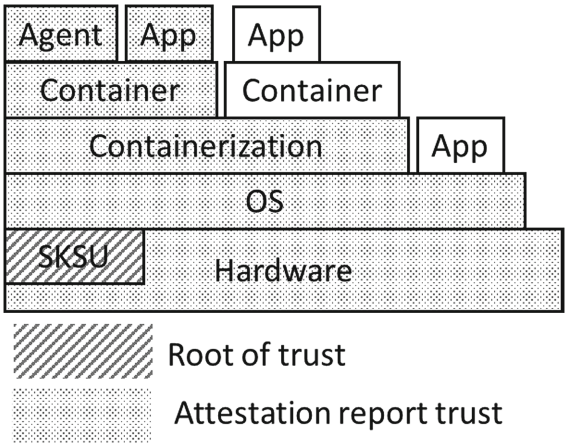

Fig. 2. Extending trust to other hardware and software using a trusted attestation report.

The external entity must be configured to expect and then validate an attestation report for an agent request. The agent's credential is stored on the TPM or other SKSU module. Such a credential alone is not sufficient for ELS authentication, because rogue software may have compromised the device and used the agent key. To secure against this attack, the attestation report validates that the proper software is installed and running at the time of the communication with the agent.

The SKSU module itself may be compromised, which would allow an attacker to generate valid attestation reports for a compromised device. This is addressed by choosing hardware devices that protect against such attacks. Such hardware is becoming a standard part of mobile phones, and keys generated on such devices are very difficult to extract (Apple 2018; Trusted Computing Group 2016).

The full secure communication sequence from agent to external entity is shown in Fig. 3 (Foltz and Simpson 2019). The steps are as follows:

(1) The agent requests an attestation report from the SKSU module.
(2) The SKSU module validates the hardware.
(3) The SKSU module validates the operating system version, configuration, and hash.
(4) The SKSU module validates the containerization mechanism or other isolation mechanism(s), if applicable.
(5) The SKSU module validates the container or other isolation unit where the agent is located, if applicable.
(6) The SKSU validates other applications in the same container as the agent.

(7) The SKSU validates the agent itself.
(8) The SKSU provides the attestation report to the agent.
(9) The agent initiates a secure connection to the external entity and validates the external entity credentials.
(10) The external entity requests authentication of the agent.
(11) The agent requests a private key operation for the agent key stored in the SKSU.
(12) The SKSU returns the results of the private key operation.
(13) The agent uses the private key operation to authenticate to the external entity and provides the attestation report through the secure connection.

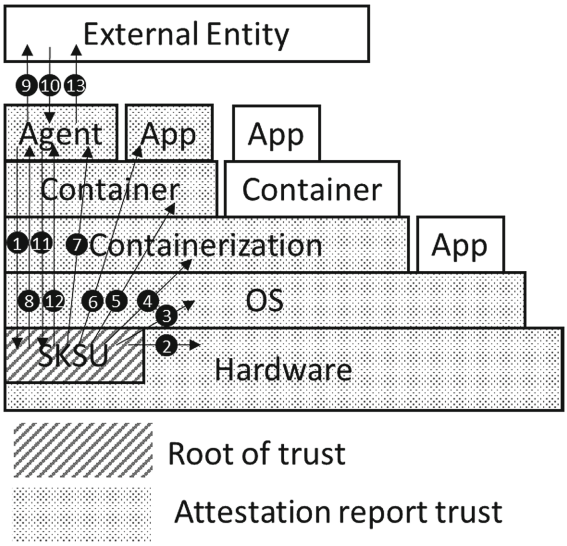

Fig. 3. Agent communication security flows.

The external entity must validate that the attestation report has a valid signature from a trusted source and that the items listed for the device conform to a valid configuration of the device. At this point, the agent has successfully authenticated to the external entity using the device key in the SKSU and by leveraging the SKSU and its internal key as a root of trust.

The external entity may then request an access token, or it may check the identity of the agent against an ACL for authorization. This process proceeds similar to normal ELS SAML requests. The only difference is that authentication to the token server also uses the flows above to use the SKSU and its attestation report for authentication.

The actions described in this section are often integrated with a mobile device manager (MDM) or unified endpoint manager (UEM). Such a system includes a central control panel and agents that reside on each endpoint device. The agents communicate with the MDM or UEM control panel and either provide data from the device or apply commands to the device. The agents leverage operating system interfaces to apply remote operations to the device. These may or may not require user acceptance or confirmation.

4 Endpoint Functionality Agents

This section discusses the agents on the endpoints that provide additional functionality. These leverage the security provided by the basic management agents in Sect. 3 and include added functionality through additional agents. Unlike the agents that help to secure the device, the agents in this section rely on that security and build on it.

4.1 Monitoring Agents

With the end-to-end security of ELS it is not possible to directly monitor the content of communication between endpoints. This information must be collected from the endpoints using agents. These monitoring agents operate on servers as well as user devices. The monitoring agents watch for potentially malicious inputs and outputs, much like a network-based monitoring system does. However, the monitoring agents only process a single device's communications. This can help performance by distributing the load across all enterprise devices. With this distributed approach, the endpoints must share some data with a central entity to enable cross-device correlations. The agent is responsible for communicating with the central aggregator and sending relevant data periodically or upon request. The agent also responds to configuration changes pushed from the central aggregator in response to changing monitoring needs. Such communications may be sent using the endpoint device management agents.

The monitoring agents process security sensitive information related to device, operating system, or application anomalies and compromises, and they initiate the transmission of this as active entities, so they must be authenticated much like the endpoint device management agents. The monitoring agent keys are stored in the TPM and used to initiate TLS connections to central servers. The agent authenticates using its key, which is coupled to an endpoint device management agent's attestation report that certifies the operational state of the device. Because monitoring agents and endpoint device management agents are both part of the standard ELS infrastructure, such attestation reports can be shared among the backend servers through a standard interface.

With a TPM attestation report from the endpoint device management system, the device's state is established as "clean." Such a clean device can then be trusted to authenticate and provide proper information from all of the agents covered by the attestation report, including the monitoring agent. The monitoring agent then provides further information about potentially malicious activity on the device itself. This information can include details of malicious operating system configuration changes, such as rooting, or malicious or anomalous application activities, such as accessing or requesting resources that are restricted.

4.2 Log Aggregation Agents

Log aggregation agents periodically assemble the relevant log content from the device, which may include monitoring logs, browser history, key usage, location history, network utilization rates, or other information as configured by the enterprise. They then send this information to an aggregator, which may further aggregate it at the enterprise level. The log information from a single device is packaged as a signed message that can be

passed through multiple aggregators without loss of security properties. The intermediate aggregators are not active entities because they do not modify the data packages. They only provide performance benefits, such as load balancing or aggregation of data packets.

Log records can come directly from the hardware. These are often handled and made available through the operating system. To ensure the log content was generated by the hardware itself, an attestation report is a natural security measure. This log attestation report is simply a digital signature that only the hardware can generate. Log records can also come from the operating system, which is often tightly coupled with a SKSU module—again, the attestation report is a natural choice for security.

Application-layer logging is more challenging. The application may not have direct control of the log files it generates. The operating system may interfere with the log file management, make log files available to other applications, or directly modify log files. The operating system could also act on behalf of the application when requesting logging related activities. Again, the attestation report for the software on the device provides a method to secure against a modified or compromised operating system. The system attestation report combined with the log attestation report provides the needed security for transferring the log record to the central aggregator.

The log aggregator has a unique position. It is a passive entity with respect to the content of the log records. These are signed by the log aggregation agents on individual devices, so such content cannot be modified by the aggregator. However, the aggregator does have an important active role to play in validating the integrity of the signature. The aggregator must validate the attestation report for the device that signed the log record. A bad attestation report implies that the signature cannot be trusted, and the log aggregator is the point where this is checked. The log aggregator signs valid log records and refuses to sign invalid log records. The aggregator serves as an active entity in providing its own validation but a passive entity with respect to the signed log record content. Each log record is treated as a blob with no internal structure. This permits the use of encryption on log records without affecting the aggregators.

The central aggregator need not be a central point of failure for log record security. Confidentiality is difficult to provide due to the nature of the aggregator, but integrity is often more important for log-related applications. The signatures from the device-based keys and certificates, combined with a validation of their attestation reports, provides a high level of integrity for such records. For aggregation functions, it may be necessary to strip the signatures and use the raw data for further processing. In this case, there is no direct method to validate the processed data, but because all original data is signed, it is possible to independently validate such computations. Thus, the central aggregator is a single point of aggregation, but it is not a single point of integrity vulnerability due to the device signatures for individual records.

4.3 Service Desk Agents

Another type of agent is installed for the enterprise service desk. Such an agent provides remote access and capabilities for a service desk person or automated service. The service desk agent provides a higher degree of access than other agents. The service desk operators often need to explore and experiment in order to troubleshoot an issue, which requires privileged access to many functions on the device. The service desk

agent, as a highly capable agent, introduces a potentially dangerous interface into the device and a tempting target of attack.

The security goals are slightly different for the service desk agents than for other agents. For other agents, the goal is strong validation of what comes out of the agents. Log records must be validated, and monitoring information must be accurate. Even attestation reports themselves must be protected. Although it is not feasible to modify the contents of an attestation report, blocking its transmission or invalidating the signature on a report with valid data could cause confusion or incorrect behavior.

For the service desk agent, the goal is strong validation of what goes into the device. It is important to prevent intruders from using the service desk agent as an attack vector into the machine. A command to reconfigure the device, if not properly validated, could put the device into an insecure state. Although such a change should be detected by the next attestation report, there is a window of opportunity for an attacker to have remote access to a device. Other agents must provide security in order for external entities to accept them and provide services. The service desk agent must strongly validate all incoming requests in order to protect the device's hardware, software, and data from malicious external entities.

The attestation reports collected by the endpoint device management system identify devices that are out of compliance. Agents will not be able to authenticate to external servers under these conditions, just as for any other agent on the compromised device. However, a service desk agent on an out-of-compliance device can potentially open the door for attackers, so a stronger response is required. Instead of just denying the service desk agent external access, the agent must be locked down or disabled until the device is brought into compliance. In order to shorten the window of opportunity for an attacker, periodic heartbeat messages can be employed by the endpoint device management agent.

The security for a service desk agent-based attack has two levels. The first is the ability of the mobile device management agent to lock the device until it comes into compliance. This could be a simple as preventing network communications, which forces a user to return the device for physical inspection and fixes. It could also remotely reset the device to factory settings. This is a very flexible and targeted response.

In situations where the attack quickly compromises the service desk agent and then the endpoint management agent, or in cases where the endpoint management agent is targeted in order to open up service desk agent interfaces, such a response is inadequate. In this case, the second level of security is the requirement by enterprise external resources of a valid attestation report from the device. After such a compromise, the attacker may control the device, but it will not be able to connect it to any other enterprise resources. This effectively isolates the device from the enterprise. This response is blunter and less effective at preventing data loss. It serves as a back-up when it is not possible to stop an attack from compromising the device. This aligns with the tenet that the enemy is among us, and the general assumption that no element is completely safe, and we must operate even when things fail.

4.4 Import and Mediation Agents

Import agents are used to refresh data in reference stores and mediate their content for compatibility with other information. The agents pull data through a guard for integrity

and accuracy checking. Guarded and filtered inputs are aggregated. Because numerous errors and inconsistencies may exist, the guard checks for formatting errors, discrepancies between data bases, incorrect or missing data, illogical data, and other undesirable conditions. Handling of discrepancies from sources depends upon the nature of the discrepancy, and corrections may be required before the data can be imported.

Import and mediation agents handle sensitive personal data that is used across the enterprise for security decisions, so they also have special responses beyond a normal agent. Any attestation report anomaly related to the import and mediation agent must lead to failure of authentication and disabling of these agents, much like the service desk agents. However, the data managed by these agents must also be rolled back to a prior known good state, because data modifications made from an import and mediation agent on a non-compliant device could have widespread lasting effects on the entire enterprise.

4.5 Self-help Agents

Self-help agents are provided on the standard desktop and provide the user with a tool to examine configuration and software conflicts. They also allow support personnel at the enterprise service desk to take over the desktop or device for diagnosis and repair of common software problems. This agent combines the capabilities of the endpoint device management agent, service desk agent, and monitoring agent. It is more common on desktops and laptops than mobile devices, but the capability can apply to any endpoint device.

Security of self-help agents has different goals and threats for different activities. Providing a user information from a static file requires very little associated security. In many cases, this may be enough to solve a problem. In other cases, the self-help agent scans the device, operating system, applications, or configurations in order to make an assessment and suggest remediation options. Because this agent has access to potentially sensitive system information, it must be included in the attestation report generated by the endpoint device management agent. This ensures that the self-help agent has not been corrupted to provide sensitive data to an unauthorized entity.

A self-help agent that not only assesses a situation but also takes action to correct it often has privileged device access. The attestation report that validates the self-help agent can also provide protection against malicious modifications by validating the self-help agent software. When the self-help agent allows remote administration of the device, it must include the protections provided by the service desk agent. This includes strong authentication of all incoming requests before taking corresponding actions on the device.

4.6 Embedded Agents

The embedded agent is for middleware, such as Java, .NET, or messaging systems, and it monitors the performance associated with application resources. This is a low level agent that is specific to its middleware. It provides information about the middleware that an application uses. This is above the layer of the operating system and below the layer of the application. It may be important for diagnosing middleware issues that do not show up in the application or operating system. For example, an application could be

running slowly but only using 40% of the available CPU despite having multiple threads. An embedded Java agent could resolve whether the Java virtual machine is limiting the CPU resources to the application or the OS is limiting the Java virtual machine.

The embedded agent should be configured to provide performance, connectivity, and anomaly data to the log file for its associated middleware. It is unaware of some of the events transpiring within the application that is built on the middleware. It can be configured to provide alerts to users or administrators. The native device's alerting system can be used for user alerts, but administrator alerts rely on approaches similar to the monitoring agent and log agent. It may be possible to integrate the embedded agent with monitoring and logging agents.

4.7 Other Agents

The preceding descriptions of agents focused on enterprise agents. These are installed on devices as part of normal enterprise operations in order to conform to enterprise rules for security and functionality. In addition, there may be other application specific agents that are desired for subgroups of the enterprise or individuals within the enterprise. These may or may not have enterprise approval or support.

Such agents can operate like the monitoring or logging agents. They ultimately rely on device hardware key storage, the operating system, and the MDM system to bootstrap the security of their communications. They require an attestation report, a hardware-based authentication key, and possibly an access token, much like any other active entity in the enterprise.

The response to an attestation report showing an out-of-compliance device depends on the function of the agent. In many cases, the response is to prevent such agents from authenticating to external servers. This is a simple and effective way to prevent the device from providing bad data or invoking sensitive enterprise services. In other cases, a honeypot approach could provide more information about the anomaly. For example, a simple bit flip in a data item would invalidate a digital signature, but it might be caused by inconsistent hardware, a software bug, or other mistakes instead of malicious activity. By accepting and recording such attestation reports in a honeypot, it is possible to perform forensics that might help to resolve mistakes while also blocking malicious activity. In addition, if it does end up being malicious, the normal systems are protected, and forensics can focus on the attack mechanics to better stop future attacks.

5 Conclusions

Moving from a centralized network-based security model to a distributed endpoint-based model provides many benefits for the current enterprise information sharing network dominated by mobile devices. However, the endpoint-based model requires careful planning to preserve existing security properties within an enterprise while adding additional functionality.

This paper explores the use of agents within the ELS model. ELS adheres to high level design principles, which are further refined to concepts and requirements. The use of agents is designed to adhere to the design principles, concepts, and requirements

to provide their functionality. The first agent must establish that the device is clean. This endpoint management agent ties directly with the operating system and hardware components. It collects attestation reports, generated by trusted device hardware, and provides them to the enterprise while adhering to ELS rules for communication. The agent software itself cannot be guaranteed to be genuine, so the software agent's role is only to request and provide the attestation report, which is a self-validating record based on the SKSU signature.

Other agents leverage the fact that the device is secure to provide their functionality. Building on this secure base, they can provide additional data, perform computations, or enable remote access to device functions. Using such an approach, the end-to-end security between all active endpoints is preserved, and network-based monitoring is performed on the devices.

This provides a way to extend the enterprise footprint onto mobile devices outside the enterprise while maintaining security comparable to internal networks.

This work is part of a body of work for high-assurance enterprise computing using web services (Foltz and Simpson 2016a, c; Simpson and Foltz 2016).

6 Extensions

Other tools or applications that use agents may use the same process to provide secure device-based communication. For example, in addition to an MDM, it is possible to use a mobile application manager (MAM) from a different vendor. The MAM has a lower level of control than the MDM due to the restricted operating system interfaces. However, it would use the same basic communication methods with external servers and internal operating system components and hardware elements.

Many mobile device applications have tight ties to external servers and serve mainly as a user interface to web APIs. Such applications function much like agents because they are lightweight and communicate with a central server. As such, the architecture described in this paper also serves as a blueprint for such applications.

References

Apple. iOS Security, iOS 12.1, November 2018. https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

Assink, A.: The Potential of Agent Architectures (2016). https://dzone.com/articles/the-potential-of-agent-architectures

Berkovits, S., Guttman, J.D., Swarup, V.: Authentication for mobile agents. In: Vigna, G. (ed.) Mobile Agents and Security. LNCS, vol. 1419, pp. 114–136. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-68671-1_7

Foltz, K., Simpson, W.: Secure endpoint device agent architecture. In: Proceedings of 21st International Conference on Enterprise Information Systems (ICEIS 2019), Heraklion, Greece, 3–5 May 2019

Foltz, K., Simpson, W.: Enterprise level security with homomorphic encryption. In: Proceedings of 19th International Conference on Enterprise Information Systems (ICEIS 2017), Porto, Portugal, 26–29 April 2017

Foltz, K., Simpson, W.R.: The virtual application data center. In: Proceedings of Information Security Solutions Europe (ISSE 2016), Paris, France (2016a)

Foltz, K., Simpson, W.R.: Enterprise level security – basic security model. In: Proceedings of the 7th International Multi-Conference on Complexity, Informatics, and Cybernetics: (IMCIC 2016). Orlando, FL (2016b)

Foltz, K., Simpson, W.R.: Federation for a secure enterprise. In: Proceedings of the Twenty-first International Command and Control Research and Technology Symposium (ICCRTS 2016). London, UK (2016c)

Liu, Z., Wang, Y.: A secure agent architecture for sensor networks. In: Proceedings of the International Conference on Artificial Intelligence, IC-AI 2003, Las Vegas, Nevada 2003

Šimo, B., Balogh, Z., Habala, O., Budinská, I., Hluchý, L.: Architecture of the Secure Agent Infrastructure for Management of Crisis Situations. Institute of Informatics, Slovak Academy of Sciences, Dúbravská cesta 9, 845 07 Bratislava, Slovakia (2009) http://www.secricom.eu/images/articles/UISAV_simo_final.pdf

Simpson, W.R.: Enterprise Level Security – Securing Information Systems in an Uncertain World, p. 397. CRC Press, Boca Raton (2016)

Trias, E.D., et al.: Enterprise level security. In: Proceedings of the 35th MILCOM Conference, pp. 31–36 (2016). https://doi.org/10.1109/milcom.2016.7795297. http://ieeexplore.ieee.org/document/7795297/

Trusted Computing Group. TPM 2.0 Library Specification, 29 September 2016. https://trustedcomputinggroup.org/resource/tpm-library-specification/

Varadharajan, V., Foster, D.: A security architecture for mobile agent based applications. World Wide Web **6**, 93 (2003). https://doi.org/10.1023/A:1022360516731

Wang, A.I., Sørensen, C.-F., Indal, E.: A Mobile Agent Architecture for Heterogeneous Devices. Department of Computer and Information Science, Norwegian University of Science and Technology, N-7491 Trondheim, Norway (2003). https://pdfs.semanticscholar.org/874b/20fbd73f5c598c8032db0c6c9e5708bc7cec.pdf?_ga=2.107853322.929957899.1544120432-1559163387.1544120432