November 2022

SECURING DATA PAYLOADS SENT FROM A CLIENT MACHINE WITH MINIMAL USER OR ADMINISTRATOR INTERACTION

Jen Bammel

David Matteson

Kyle Mills

Claire Gower

Todd Chapman

See next page for additional authors

Inventor(s)
Jen Bammel, David Matteson, Kyle Mills, Claire Gower, Todd Chapman, and Adam Goodman

SECURING DATA PAYLOADS SENT FROM A CLIENT MACHINE WITH MINIMAL USER OR ADMINISTRATOR INTERACTION

AUTHORS:
Jen Bammel
David Matteson
Kyle Mills
Claire Gower
Todd Chapman
Adam Goodman

ABSTRACT

Techniques are described herein for providing an extra layer of security for a Multi-Factor Authentication (MFA) application (e.g., a device health application) installed on a user's machine to ensure that the data payload being sent to authentication servers came from the authenticating user's machine. The application may be enrolled with a cryptographic keypair stored in the hardware of the user's machine. The key may be used to sign health data payloads sent to the MFA servers.

DETAILED DESCRIPTION

The device health application collects data from a user's machine before MFA to determine the health posture of the machine. This data is forwarded using a POST request along with a transaction identifier (ID) that ties the health data collected by the application to the authentication itself. Beyond those two security measures, there is no way to absolutely ensure that the health check came from the device health application instance installed on that user's machine.

Accordingly, techniques are described herein to further secure the data payload sent from the health application as part of the authentication process. The authentication flow happens in stages: primary authentication (either with a username/password or with another, password-less type of authentication), then pre-authentication, and finally secondary authentication, where the user is asked to make use of a second factor to gain access to the resource. The techniques described herein focus on the pre-authentication portion of the authentication flow.

During pre-authentication, information regarding the user’s machine is collected to determine whether the user should be allowed to proceed to second factor authentication. The information may vary based on the administrator’s policy settings, but the techniques described herein focus on the “health check” sent to the MFA servers by the device health application.

The MFA prompt sends a request to the device health application, which is listening on a set of localhost ports. The application validates that the incoming request is from an MFA server, then proceeds to collect health information about the device and build up a JavaScript Object Notation (JSON) data payload to send up to a secondary MFA server. This data payload, as it exists today, has no encryption or signing, and relies solely on Hypertext Transfer Protocol (HTTP) messages to send the data up to the MFA application.

The techniques provided herein add an extra layer of security to this health data payload by registering the device health application instance on the user’s machine with the MFA servers ahead of time. This enrollment process ensures that the device health application can make use of a private key stored in the hardware of the user’s machine to generate a signature for this data payload before sending both the signature and an encrypted version of the JSON data payload up to the MFA servers.

Once the health data payload – now paired with a signature – returns to the engine behind the MFA prompt, a public key stored in the MFA servers that correlates to the user and the machine from which the user is authenticating may be used to validate the incoming signature. If the incoming signature is valid, the incoming health data payload may be accepted; otherwise, the incoming health data payload may be rejected, and a report may be sent back to the user via the MFA prompt to indicate that an error has been encountered.

The way in which the keypair is generated and stored may enable this signature generation and validation. When it has been determined that the user needs to register the device health application with the MFA servers, a process known as “enrollment” is triggered.

The prompt sends a request to the device health application localhost HTTP server on a specific enrollment route. This request is validated in a similar fashion to how the health check request is validated, meaning that the request headers are checked to ensure the origin was an MFA server. The data contained in the request is the server to which the enrollment data should be sent once enrollment is complete. The MFA-generated identifiers may also be provided so that the user's machine may be uniquely identified in the database.

The device health application may generate a cryptographic key-pair. The private key is stored in the Trusted Platform Module (TPM) on Microsoft Windows® or in the Secure Enclave on macOS to ensure that it remains tethered to the user's hardware. Finally, the public key generated during this enrollment process may be sent in an HTTP POST to the Uniform Resource Locator (URL) sent in the original enrollment request. This ensures that the data is sent to the correct server in the MFA backend so that the prompt engine knows where to watch for this data. Certificate Authority (CA) pinning and URL validation may ensure that the data is sent to an MFA server before the POST is sent. This may be similar to the way the health check works.

This enrollment flow is beneficial for both users and administrators because it does not require any additional work for the user or the administrator. Instead, the flow may be integrated into the MFA prompt at any point (e.g., the most secure point) and can seamlessly happen behind the scenes (e.g., with a small message to the user as the enrollment is taking place to update the user).

The functionality of enrollment may be locked behind a trust check. This trust check ensures the machine for which the user is attempting to enroll the device health application is registered as part of a customer's fleet. However, the general functionality of the techniques described herein do not necessarily rely on this trust restriction.

Figure 1 below illustrates an example flow diagram for securing data payloads as described herein.

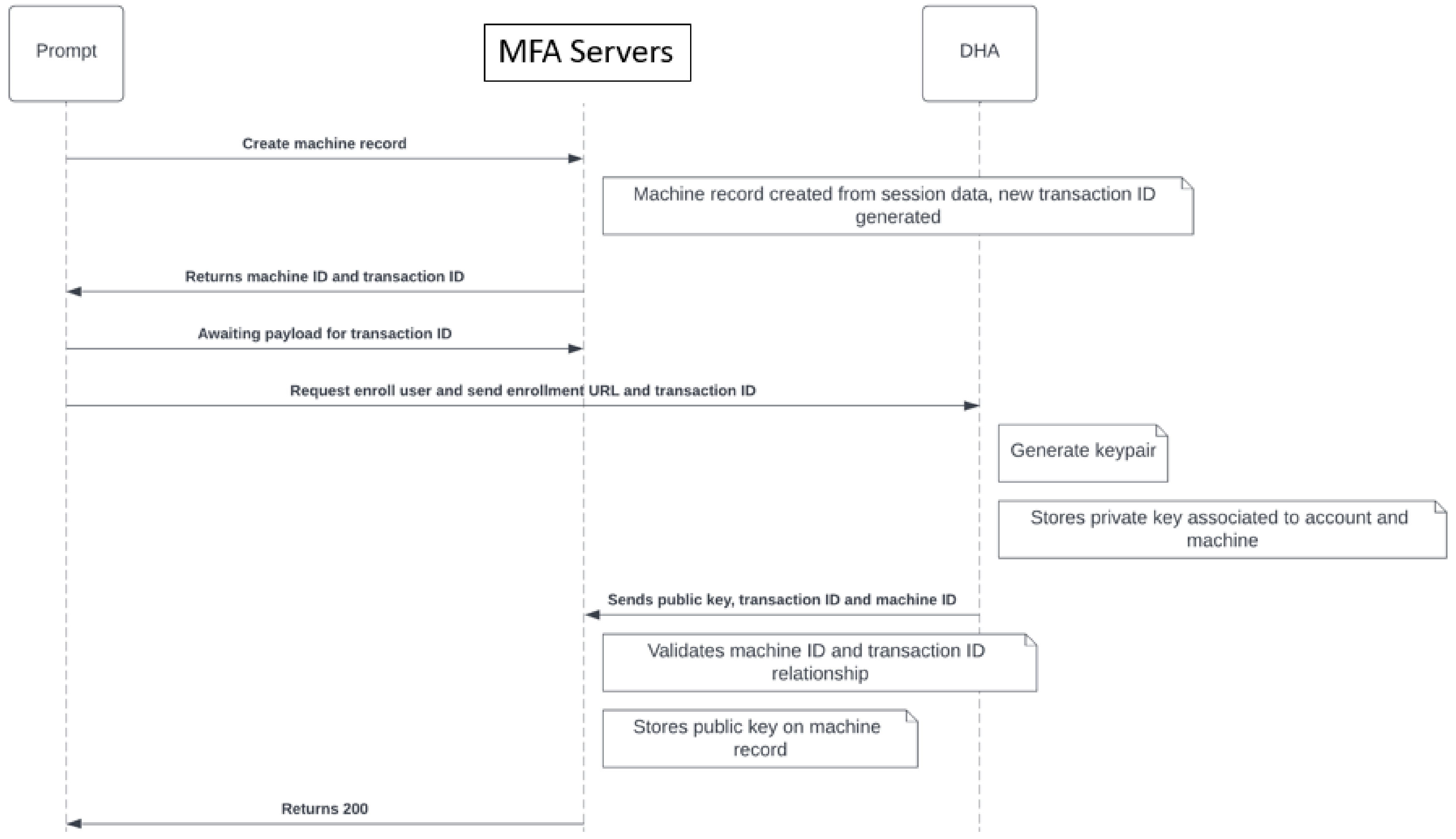
Prompt
MFA Servers
DHA
Create machine record
Machine record created from session data, new transaction ID generated
Returns machine ID and transaction ID
Awaiting payload for transaction ID
Request enroll user and send enrollment URL and transaction ID
Generate keypair
Stores private key associated to account and machine
Sends public key, transaction ID and machine ID
Validates machine ID and transaction ID relationship
Stores public key on machine record
Returns 200

Figure 1

In summary, techniques are described herein for providing an extra layer of security for an MFA application (e.g., a device health application) installed on a user's machine to ensure that the data payload being sent to authentication servers came from the authenticating user's machine. The application may be enrolled with a cryptographic keypair stored in the hardware of the user's machine. The key may be used to sign health data payloads sent to the MFA servers.