Security and Mutual SSL Identity Authentication for IoT

Author:
Sonia Burney – Solutions Architect, Akamai Technologies
Security and Mutual SSL Identity Authentication for IoT

Introduction: How We Got Here
As we advance in technology, we need to advance security measures to protect that technology. What kind of technology are we talking about? Let's start with the example of the modern web. With usage growing exponentially, companies are now required to strengthen how they protect against the large influx of traffic.

Over the years, companies across all industries have devised smarter ways to protect users as they navigate to different types of websites. Additionally, after enhancing the web experience, companies further advanced the ways in which they protect users accessing content via native or mobile web applications. As technology has further evolved, stronger countermeasures were created to prevent attacks and strengthen overall security posture. Today, these best practices have become must-haves for businesses.

The Unique Challenges of Identity in IoT
After making strides to adapt to growing security threats for web and mobile devices, what comes next? We now need to protect next-gen technology like Internet of Things (IoT) devices.

IoT devices have changed the way customers interact with technology. Instant connections and voice technology allow consumers to play music, control their homes, and more. This is made possible through devices and services such as Alexa and Echo, Google Home, Sonos home entertainment, and Sony PlayStation. These businesses and brands need innovative strategies in order to protect their users. The new security model will benefit all new browsers, applications, and IoT delivery.

Traditional methods of origin or CDN protection are not enough. Many businesses and brands must increase their adoption of IoT industry best practices, and protect devices and users when enabling experiences through IoT.

Proposed Approach for Secure Mutual SSL Authentication
Today, technologists and practitioners are establishing new and creative ways to solve for these complex security issues. Let's dive into one specific use case of incorporating a security-related feature from the web into IoT devices.

Mutual SSL authentication, or certificate-based mutual authentication, involves two separate identities authenticating one another through the use of certificates. Even if a business uses client reputation strategies and has security measures in place, the two identities can provide absolute verification.

Below is a step-by-step overview of how an enhanced security model, driven through mutual SSL authentication, can work.

• **Initial Validation:** The app, website, or device (client) must authenticate itself with the server by sending a unique certificate installed at the client. This confirms a unique identity with the server to ensure initial validation via Certificate Authorities (CAs).

• **Verifying Status:** The server queries out to the Online Certificate Status Protocol (OCSP) responder to verify the revocation status of that certificate (X.509 digital certificate). However, Certificate Revocation Lists (CRL) are also an option to leverage. OCSP was created to address some of the issues with CRL, so this document will cover OCSP responder implementations with regard to client certificate status checking.

• **CDN Layer Validation:** The responder will typically return a signed response determining whether the client certificate is deemed as good, revoked, or unknown (other status messages available); and, based on the feedback from the responder, the server can take appropriate action. At a CDN layer, this operates the same way, except that the logic to query out to the responder remains at a CDN Edge Server/Node (as opposed to going all the way to the origin to validate the certificate). Enabling authentication at the CDN versus solely relying on the origin reduces latency and the time needed to validate a user’s client certificate, especially when leveraging the CDN’s caching capabilities for different response conditions. Additionally, reducing the number of requests hitting the origin for authentication minimizes the risk for origin overload.
• **Responder Results:** At your CDN, the steps would involve the upload of relevant Certificate Authorities (CAs) as well as the OCSP CAs. At a high level, the configuration would need to consist of references to the uploaded CAs, the OCSP responder endpoints, and the handling of the certificate revocation status as a result of the responder checks. Also, if there are multiple OCSP responder endpoints mapped to one or more hostnames that share configurations, then AIA chasing should be configured to extract the responder endpoint from the client certificate itself from the AIA (Authority Information Access) field.

• **Origin Verification:** At your CDN, authentication can also be configured going forward to origin. Going forward from the CDN to the origin can involve different protection strategies. These include origin whitelisting of edge node IPs allowed to access content, or even establishing certificate validation from the edge node to the origin.

Through these steps, by enabling client certificates at the browser, we are able to protect the flow between the client and the CDN. At a high level, configuring mutual authentication at the CDN layer offloads verification, revocation, reporting, and blacklisting capabilities. Essentially, the CDN is now taking the end user’s client certificate info, validating it by submitting the request to the OCSP responder, and then taking action based on the response. That action can involve reporting via log lines, Akamai cloudlets, or other Akamai/third-party cloud-monitoring tools. Based on status received at the CDN layer, in addition to reporting capabilities, we can provide alternate pages or even deny users.

Traditionally, this has been implemented for various web applications, and most of the issues in the configuration process have been outlined and fixed. Testing and debugging issues has proven to be fairly straightforward, as we can typically dig out client certificate-related logs from the CDN or server layers, or even rely on browser tools for debugging and ensuring correctness of this type of solution.
Getting Started with an Enhanced IoT Security Model

With the introduction of mutual authentication for IoT Devices, best practices must be followed for enhanced security. Practitioners must consider configuration, testing, debugging, and more.

Top questions to consider are:

- How do we configure these devices to enforce client certificate requests?
- What does that workflow look like?
- How do we test these devices in a staging environment?
- Can we replicate configuration and testing scenarios in a virtualized environment?
- How do we debug IoT device issues?
- What is the difference in supportability between IoT devices and web applications?
- What is the difference in supportability between IoT devices and web applications?

IoT devices do not follow the standard interface that CDNs or developers are accustomed to in terms of configuring, testing, and debugging. New features and innovations require design and implementation to travel through the process lifecycle.

As mentioned, the standard best practices implementation strategies for mutual SSL authentication for the web do not work exactly as expected with IoT devices. IoT devices are unique, and for enhanced security at your CDN, there are stricter requirements. Here are the top considerations when implementing a solution at your CDN:

<table>
<thead>
<tr>
<th>Verification</th>
<th>CAs and OCSP CAs need to have the appropriate signing metadata before uploading them to the platform, where these can be verified by the platform, before checking any client certificates’ revocation status.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enablement</td>
<td>Strictest authentication should be implemented at each CDN layer and enforced completely for each hostname.</td>
</tr>
<tr>
<td>Third-Party Server Applications</td>
<td>Mutual authentication must be enforced completely (required) to avoid advertising the information publicly and enforce user behavior.</td>
</tr>
<tr>
<td>Device OpenSSL Commands</td>
<td>The physicality of these devices means API endpoints are accessed via the device server and require additional measures beyond standard web certificate provisioning.</td>
</tr>
<tr>
<td>OCSP Responder Configuration</td>
<td>The error and status messages have more variance with IoT devices. Thorough testing and mapping across statuses ensure that each failure case is remediated appropriately.</td>
</tr>
<tr>
<td>Issuing Certificates to the Origin</td>
<td>Upon successful validation of a user’s incoming client certificate, we can pass along the client certificate and subject domain name in headers going forward to the origin. There are various fields that can be extracted from the client certificate in advanced metadata, for logging purposes at both the edge nodes and the origin.</td>
</tr>
<tr>
<td>Device-Level Logging</td>
<td>This assists with revocation status checking and will help debug any issues that may come up. This could range from determining whether or not the request is even hitting the OCSP responder to analyzing how different request method/path combinations are handled by the edge nodes and OCSP responder.</td>
</tr>
<tr>
<td>Edge to Origin Verification</td>
<td>Strategic options in the workflow for edge node to origin include leveraging Akamai’s Site Shield or using certificate authentication (a longer process).*</td>
</tr>
</tbody>
</table>

*Note: The last point is a continuation of the flow and is not a separate question for top considerations.
* Also, at your CDN, you need to consider how authentication happens from an edge node to the origin. The above consists of information for client to edge node authentication. The flow from edge node to the origin can be handled in different ways, such as leveraging Akamai’s Site Shield to whitelist IPs for controlled access, or even using certificate authentication going forward to the origin. This process, although longer, is similar to the process of configuring client certificate validation from client to edge node.

**Designing with Akamai**

There are various design and implementation strategies to consider when enforcing mutual SSL authentication across hostnames. Depending on your requirements and current architecture, the design can go from simple to complex. Often, businesses seek expert practitioners to guide the way to properly scope the initiative and build accountability for any potential short-term workarounds.

Akamai has over a decade of experience enabling client certificate authentication for browsers and web-based clients, and supports mutual SSL authentication on our platform for IoT today. This ensures the addition of mutual SSL authentication can work at scale for our customers’ IoT devices that are enabled, creating seamless customer experiences across the globe.

To ensure your business is following IoT best practices, scope an IoT mutual SSL engagement, or just simply ask us questions, reach out to consulting@akamai.com.

**About the author**

Sonia Burney is a Solutions Architect for Akamai Technologies with 10 years of experience in the web performance and web security spaces. Sonia has spoken at O'Reilly conferences and written a book that covers strategies to improve both security and performance from a front-end perspective. Prior to joining Akamai, Sonia worked at various companies as an experienced full-stack developer.