rules.sonarsource.com

C++ static code analysis: Server certificates should be verified during SSL/TLS connections

5-7 minutes

Validation of X.509 certificates is essential to create secure SSL/TLS sessions not vulnerable to man-in-the-middle attacks.

The certificate chain validation includes these steps:

- The certificate is issued by its parent Certificate Authority or the root CA trusted by the system.
- Each CA is allowed to issue certificates.
- Each certificate in the chain is not expired.

It’s not recommended to reinvent the wheel by implementing custom certificate chain validation.

TLS libraries provide built-in certificate validation functions that should be used.

Noncompliant Code Example

libcurl

```
#include <curl/curl.h>

CURL *curl;
curl_global_init(CURL_GLOBAL_DEFAULT);

curl = curl_easy_init();
curl_easy_setopt(curl, CURLOPT_URL,
"https://example.com/");
curl_easy_setopt(curl,
CURLOPT_SSL_VERIFYPEER, 0L); //
Noncompliant; CURLOPT_SSL_VERIFYPEER is set
to 0, no peer's SSL certificate will be verified

//Perform the request
curl_easy_perform(curl);
```

OpenSSL

```
#include <openssl/ssl.h>

const SSL_METHOD *method = TLS_method();
SSL_CTX *ctx = SSL_CTX_new(method);

SSL_CTX_set_verify(ctx, SSL_VERIFY_NONE,
NULL); // Noncompliant; SSL_VERIFY_NONE means
no automatic certificate verification
```

```
SSL *ssl = SSL_new(ctx);

// ...

SSL_connect(ssl);
#include <openssl/ssl.h>

static int verify_callback(int preverify_ok,
X509_STORE_CTX *ctx) { return 1; } // This callback
always validate the certificate

const SSL_METHOD *method = TLS_method();
SSL_CTX *ctx = SSL_CTX_new(method);

SSL_CTX_set_verify(ctx,
CURLOPT_SSL_VERIFYPEER, verify_callback); //
Noncompliant; the verify callback result overrides
OpenSSL built-in verification enabled by
CURLOPT_SSL_VERIFYPEER option.

SSL *ssl = SSL_new(ctx);

// ...

SSL_connect(ssl);
```

botan

```
#include <botan/tls_client.h>
#include <botan/tls_callbacks.h>
#include <botan/tls_session_manager.h>
#include <botan/tls_policy.h>
#include <botan/auto_rng.h>
#include <botan/certstor.h>
#include <botan/certstor_system.h>

class Callbacks : public Botan::TLS::Callbacks
{
// ...

virtual void tls_verify_cert_chain(
        const std::vector<Botan::X509_Certificate>
&cert_chain,
        const std::vector<std::shared_ptr<const
Botan::OCSP::Response>> &ocsp_responses,
        const std::vector<Botan::Certificate_Store *>
&trusted_roots,
        Botan::Usage_Type usage,
        const std::string &hostname,
        const Botan::TLS::Policy &policy) override {} //
Noncompliant (secondary location),
tls_verify_cert_chain never throws. Always accept
server certificate and doesn't verify hostname
};

class Client_Credentials : public
Botan::Credentials_Manager
{
// ...
};


Callbacks callbacks;
Botan::AutoSeeded_RNG rng;
Botan::TLS::Session_Manager_In_Memory
session_mgr(rng);
Client_Credentials creds;
Botan::TLS::Strict_Policy policy;

// open the tls connection
Botan::TLS::Client client(callbacks, session_mgr,
creds, policy, rng,

Botan::TLS::Server_Information("example.com",
443),

Botan::TLS::Protocol_Version::TLS_V12); //
Noncompliant; uses an implementation of
```

Botan::TLS::Callbacks that doesn't validate server certificate

Compliant Solution

libcurl

```
#include <curl/curl.h>

CURL *curl;
curl_global_init(CURL_GLOBAL_DEFAULT);

curl = curl_easy_init();
curl_easy_setopt(curl, CURLOPT_URL,
"https://example.com/");
curl_easy_setopt(curl,
CURLOPT_SSL_VERIFYPEER, 1L); // Compliant;
CURLOPT_SSL_VERIFYPEER is set to 1

//Perform the request
curl_easy_perform(curl);
```

OpenSSL

```
#include <openssl/ssl.h>

const SSL_METHOD *method = TLS_method();
SSL_CTX *ctx = SSL_CTX_new(method);
```

```
SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, NULL); // Compliant; CURLOPT_SSL_VERIFYPEER enable OpenSSL's built-in verification of the peer certificate.

SSL *ssl = SSL_new(ctx);

// ...

SSL_connect(ssl);
```

botan

```
#include <botan/tls_client.h>
#include <botan/tls_callbacks.h>
#include <botan/tls_session_manager.h>
#include <botan/tls_policy.h>
#include <botan/auto_rng.h>
#include <botan/certstor.h>
#include <botan/certstor_system.h>

// Compliant use the default implementation of tls_verify_cert_chain method which verify server certificate and hostname
class Callbacks : public Botan::TLS::Callbacks
{
// ...
};
```

```
class Client_Credentials : public
Botan::Credentials_Manager
{
// ...
};

Callbacks callbacks;
Botan::AutoSeeded_RNG rng;
Botan::TLS::Session_Manager_In_Memory
session_mgr(rng);
Client_Credentials creds;
Botan::TLS::Strict_Policy policy;

// open the tls connection
Botan::TLS::Client client(callbacks, session_mgr,
creds, policy, rng,

Botan::TLS::Server_Information("example.com",
443),

Botan::TLS::Protocol_Version::TLS_V12); //
Compliant; uses an implementation of
Botan::TLS::Callbacks that validate server certificate
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2021 Category A7 - Identification and Authentication Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- Mobile AppSec Verification Standard - Network Communication Requirements
- OWASP Mobile Top 10 2016 Category M3 - Insecure Communication
- MITRE, CWE-295 - Improper Certificate Validation