Constrained CIP Security Best Practices

Jakub Korbel, Jack Visoky
Rockwell Automation
Why the CIP Security Constrained Profile?

• More devices are being “connected” via Ethernet
  – Even sensors and other constrained devices
• Each of these present an attack surface that might be attractive to hackers
  – Example: Mirai Botnet
• Even if a device isn’t directly connected to the Internet it could still be targeted
  – Example: Jeep hack, Trisis malware
• Therefore security protections must be available for resource-constrained CIP devices
Resource Constrained CIP Security

• In April of 2021 ODVA published the CIP Security Resource-Constrained Profile
• Provides functionality for protocol protections, but tailored to more constrained devices
  – PSKs instead of certificates (except for initial commissioning)
  – Only 3 cipher suites
  – DTLS only
• This makes it much easier for a constrained device to implement security
• However, there are still some best practices to discuss
Why the changes – certificates

• Certificates bring a lot of complexity
  – X.509 parsing
  – File Object
  – Certificate Management Object
• Using PSKs is a lot simpler, better suited to these constrained devices
• However, an initial, default certificate can still be used – no need to parse it or support the objects, just serve it as part of the initial handshake
  – This helps bootstrap security and to provide device authenticity assurances
  – IEEE 802.1AR IDevID

Part of a certificate used on an EtherNet/IP device...seems complicated...

X509 Certificate:
Certificate Extensions: 13
Version: 3
Serial Number: 6e6e5112000000000f24
Signature Algorithm:
  Algorithm ObjectId: 1.2.840.10045.4.3.4 sha512ECDSA
  Algorithm Parameters: NULL
Issuer:
  CN=Rockwell Automation - Manufacturing Intermediary CA
  O=Rockwell Automation, Inc.
  C=US
  Name Hash(sha1): 02a938c1873ccc4c6c926b3a1e3a2080b3678e1f
  Name Hash(md5): 3ebb00acfb867ffe59d9ac3fe3a3
NotBefore: 12/16/2015 12:19 PM
NotAfter: 12/6/2055 12:19 PM
Subject:
  CN=1756-L85E (00b48f94)
  O=Rockwell Automation, Inc.
  C=US
  Name Hash(sha1): ec91018dbb124274ba596b0f46e7a939a65bc36
  Name Hash(md5): 31c2a3931fd0ce674b2306e07e3a76
Public Key Algorithm:
  Algorithm ObjectId: 1.2.840.10045.2.1 ECC
  Algorithm Parameters:
  06 08 2a 86 48 ca 3d 03 01 07
  1.2.840.10045.3.1.7 ECDSA_P256
Public Key Length: 256 bits
Public Key: UnusedBits = 0
  0000 02 01 01
  0000 02 01
  1.3.6.1.4.1.95.3.2: Flags = 0, Length = 3
  0000 02 01
Name Hash(sha1): ec91018dbb124274ba596b0f46e7a939a65bc36
Name Hash(md5): 31c2a3931fd0ce674b2306e07e3a76
Public Key Algorithm:
  Algorithm ObjectId: 1.2.840.10045.2.1 ECC
  Algorithm Parameters:
  06 08 2a 86 48 ca 3d 03 01 07
  1.2.840.10045.3.1.7 ECDSA_P256
Public Key Length: 256 bits
Public Key: UnusedBits = 0
  0000 04 18 ee 09 3c fd 6e cb 31 13 0 17 32 43 db
  0010 7b b0 3a 55 47 61 f7 a4 2e b5 9d 02 88 a5 d2 e8
  0020 12 3a 73 d9 38 f6 46 ce 37 35 fa 11 aa ea 2f 20
  0030 cc 27 21 38 d6 78 be 44 9f df 2b 6e 1e 91 96 73
  0040 4f
# Why the changes – ciphers

Just 3 ciphers needed for the Resource Constrained Profile

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_ECDHE_PSK_WITH_NULL_SHA256</td>
<td>Authenticity only, no confidentiality. Just uses the SHA-256 HMAC</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>AES GCM for authenticity and confidentiality. Highly optimized for hardware acceleration</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>ChaCha20 and Poly1305. Highly optimized for devices without hardware acceleration</td>
</tr>
</tbody>
</table>
Why ECDHE Ciphers?

- Note that each cipher is Elliptic Curve Diffie Hellman Ephemeral
- This means that an ephemeral key pair is generated during the handshake
- This does add cost/complexity in terms of code space and memory usage 😞
- However, it also provides perfect forward secrecy 😃
  - Prevents compromise of data transmitted if static keys are ever compromised
- SIG felt this was important enough to pay the cost, minimum for security
- However it is something that could be discussed in the future if further constraints are needed

PERFECT FORWARD SECRECY!!

It's worth the memory usage (until it's not...)
But How?

• The CIP Security Resource-Constrained Profile defines the technologies, but does not provide guidance on implementation details

• Here are some practices discovered and established during Rockwell Automation’s implementation of the profile
Security Implications

• Trade-offs come at a cost!
  – Pay attention to your threat model

• Certificates are complex, but they provide a unique identity for every device in the system
  – PSKs don’t
What are some downsides to PSKs?

- Non-repudiation – anyone with the PSK can deny they did something
- Spoofing – cannot distinguish one member with the PSK from another
- Reflection – in some cases connections can be reflected or redirected
- Note with sufficiently small groups (like of 2) many of these threats disappear, but that is often not the case when using PSKs
Tasking

• Cooperative multitasking == tasks are relinquishing CPU control themselves, single stack is sufficient
  – Did someone say “event loop”?
• Pre-emptive multitasking == scheduler relinquishes task’s CPU control usually as a reaction to ISR (context switch after finished assigned time slice), a stack per task is needed
  – Did someone say “POSIX threads”?
Pre-emptive Multitasking

WE’VE HAD ONE, YES

WHAT ABOUT SECOND STACK?
Tasking

- Task architecture has a big impact on the performance and footprint of security in a resource-constrained environment.
- Cooperative multitasking can be suitable if developers have full control over running firmware.
  - With single core and well-behaving tasks finishing operations in small chunks, Cooperative multitasking can present significant performance benefit over pre-emptive multitasking.
- Pre-emptive multitasking is a must when users can launch misbehaving tasks, so that the system won’t halt. It also does not require developers to split large calculations or allows them to perform blocking operations.
What is a DTLS Handshake?

Client

ClientHello

HelloVerifyRequest [cookie]

ClientHello [cookie]

ServerHello

(Certificate)

ServerKeyExchange

(CertificateRequest)

ServerHelloDone

(Certificate)

ClientKeyExchange

(CertificateVerify)

ChangeCipherSpec

Finished

ChangeCipherSpec

Finished

Application Data

Server
Handshake Stack and Heap consumption

- Handshake and also hash/crypto operations can be very **stack and heap-consuming**, check it out yourself here.
  - Handshake consumes more stack/heap memory than keeping a connection alive
  - DTLS consumes more memory than TLS, but remember TLS leverages connection and stream properties of TCP
  - ECC client side is more memory consuming than RSA, but this might change with certificate chains, where the sum of key sizes offsets ECC's stack and heap-heavy operations
  - Even with size tweaking compile-time options, the stack sizes just for handshake grew often to 10K
Handshake operations are also performance intensive, especially for ECDHE.

CIP I/O cannot wait till the Ephemeral key generation (RNG) of ECDHE finishes.

- **Pre-emption** can be the solution, but it can result in multiplying RAM consumption for task stacks.
- **Splitting the calculations** can be the solution in Cooperative scheme, but it requires support in the TLS library.
- **Hardware acceleration** can help in combination
  - More and more cryptochips are being made, some can even help ECC operations.
Best practices

• To get to smaller memory footprint and better performance in highly constrained environments, such as <256KiB of RAM and <300MHz CPU Speed:
  – Use cooperative multitasking.
  – Use cryptoacceleration, if available.
  – Prefer PSKs, but understand the impact on security and establishment of trust.
  – Benchmark (or read the documentation of) cryptographic libraries for memory footprint and handshake performance.
  – Pick cipher-suites that provide enough security and can sustain the traffic needed by your product function.
    • (Or just conform to Volume 8, CIP Security Resource Constrained Profile 😊)
Conclusions

• CIP Security Resource Constrained Profile provides robust security while fitting into tightly constrained environments
• It is important to be aware of security limitations with PSKs
• Task architecture really matters for constrained environments and security, think about how constrained your environment is and pick your tasking model
Conclusions

• Be careful of how the handshakes are done, this has a big impact on performance and memory usage

• If in practice this profile is still too burdensome to highly constrained devices the SIG can consider PSKs which don’t provide Perfect Forward Secrecy
  – This is likely a reasonable trade-off, bringing some security to constrained devices is better than “perfect” security that doesn’t get used