05 / 11 / 23 DevSecOps Days Pittsburgh

Securing the IoT Supply Chain with DevSecOps

Daniel Morrison && Antonio Escalera on behalf of Raft

RAFT

00 Objective

- Provide a look at some of the most impactful DevSecOps approaches/technologies available and explore how IoT manufacturers may strategically implement them to augment their security posture.
- NOT to provide a complete, production-ready solution.

01 Develop with Security in Mind

- Security is built from the first commit, so sign them
- Adhere to the *principle of least privilege* within your source code repositories; configure your branches and organization to protect them from bad actors
- Establish an agreed upon *definition of secure* and *threat tolerance* for ingesting open-source dependencies; keep them up to date and secure
- Consistently perform SAST to ensure accepted levels of threat are not breached

SLSA

sigstore cosign

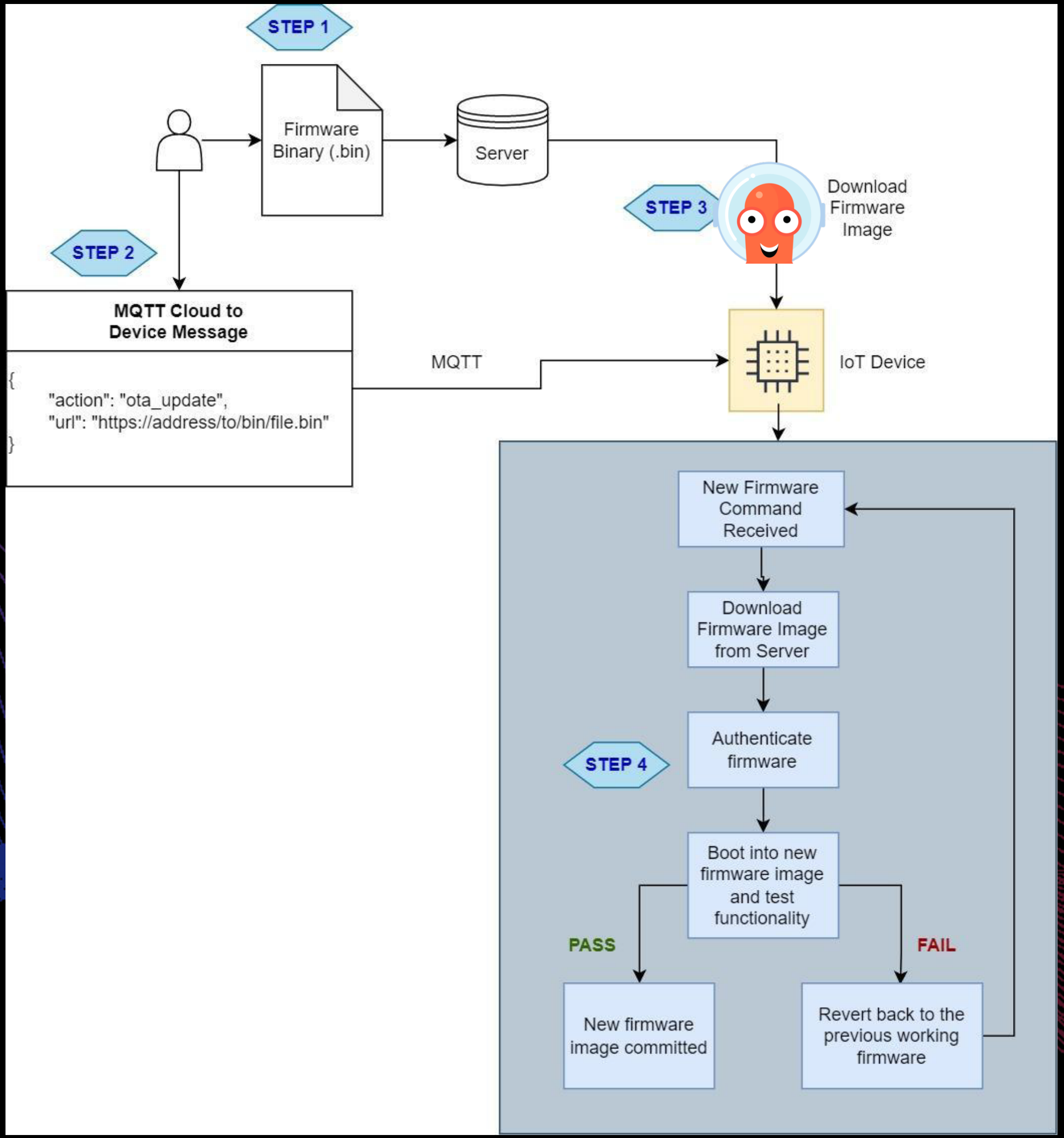
syft

02 Securing the Build Process

- *Where* and *how* you build matters (a lot), make the right choice for your Supply Chain Levels for Software Artifacts(SLSA) needs
- Generate provenance for every build which produces artifacts and make that provenance available alongside those artifacts
- All OCI images produced should be signed and be made available alongside their (attested) SBOM

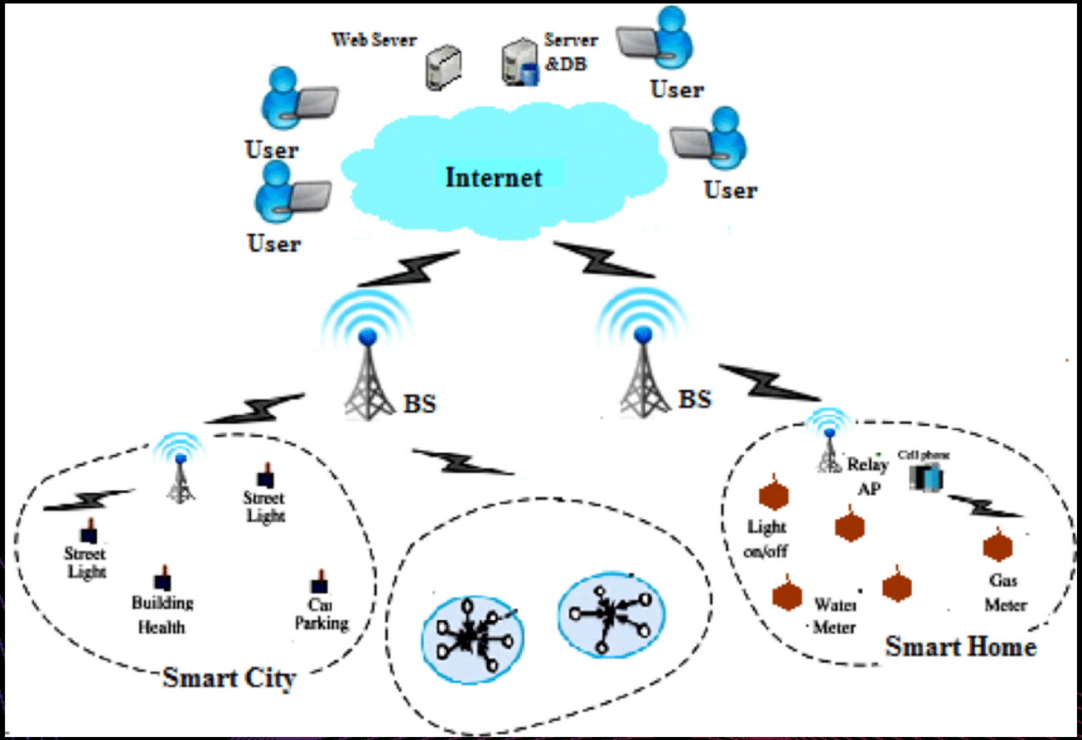
03 Update Automation && Pipelines

- GitOps for applications
- GitOps for infrastructure
- GitOps for firmware
- Sporadic Connectivity versus Always-On



04 Network Layer: WWAN && WLAN

- OTA updates, monitoring, control
- Security throughout layers
- 5G-AKA/EAP-TLS, IPSec, 802.1X
- Encryption & access-control – confidentiality & integrity
- Network segmentation/application brokers
- Traffic monitoring – NetFlow
- Continuous evaulation – OpenVAS, OpenSCAP



05 Remote Attestation && Field Registration

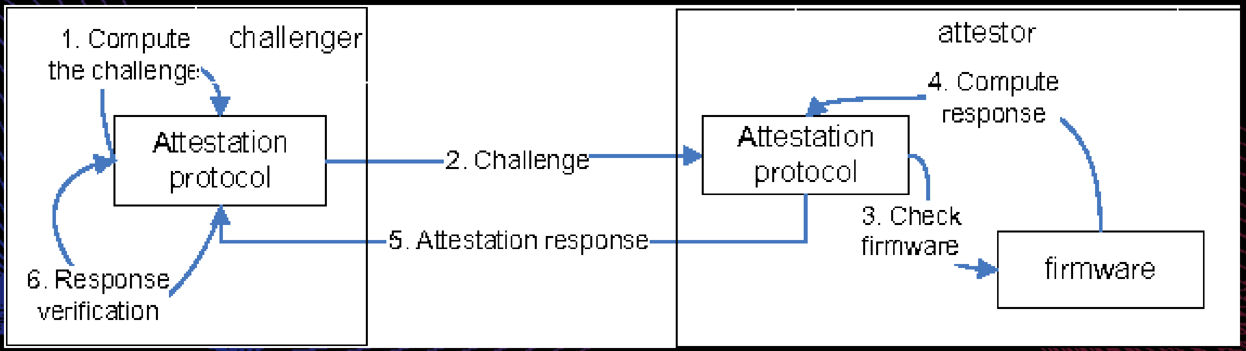
TPM-enabled Remote Attestation Protocol (TRAP)

Devices ship with:

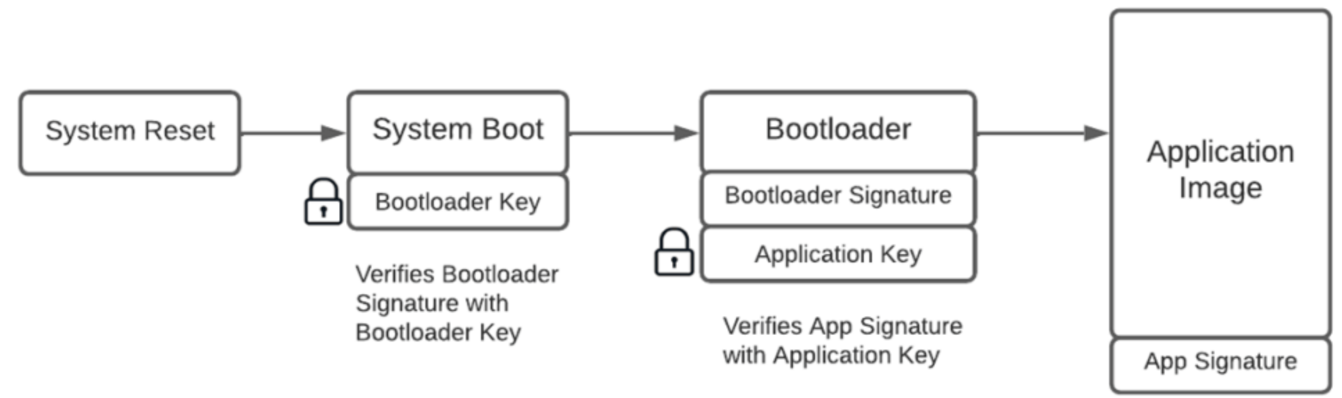
- Device-specific enrollment key
- Bootloader hash
- Printed registration code

Customer boots device:

- TPM verifies bootloader contents
- Bootloader hashes application content/rootfs
- User connects to device and inputs registration code



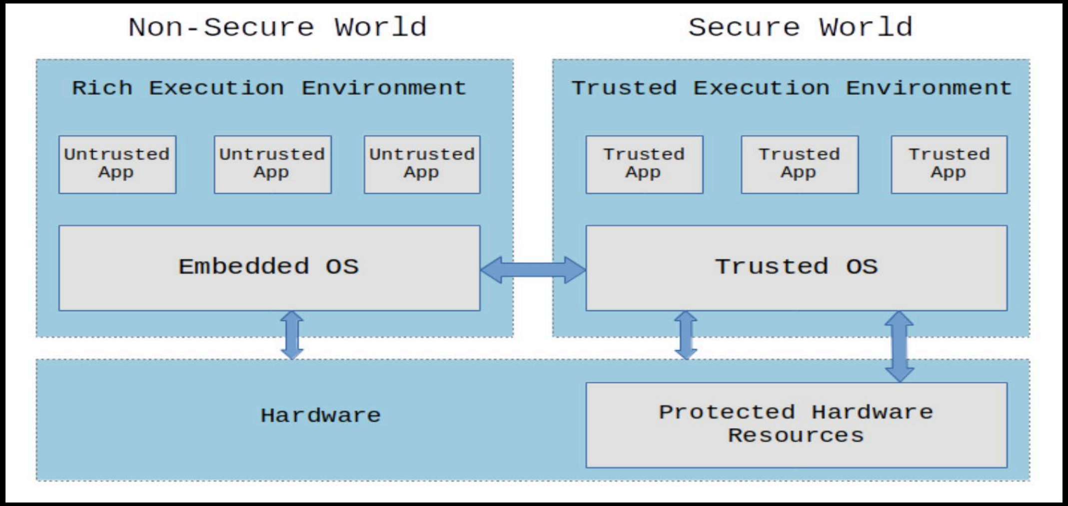
06 Secure Boot



Typical Secure Boot Software Flow

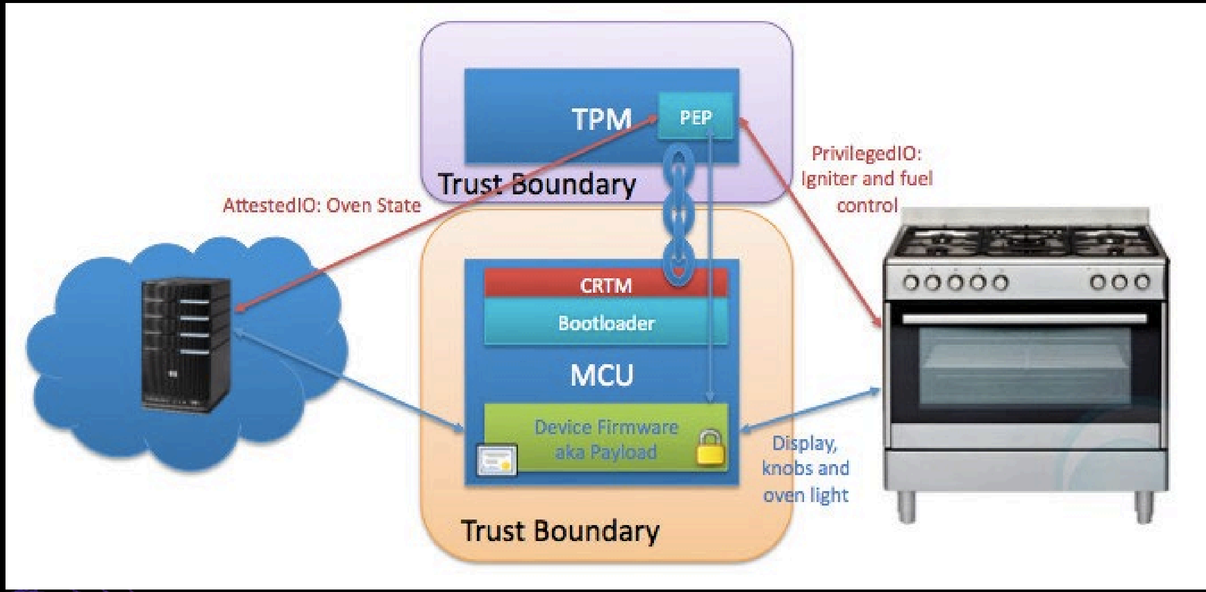
- Root of trust in secure element
- Bootloader verifies authenticity and integrity
- Signing and verification of firmware and software
- Chain of trust between components
- Updates and patching

07 Trusted Execution Environment

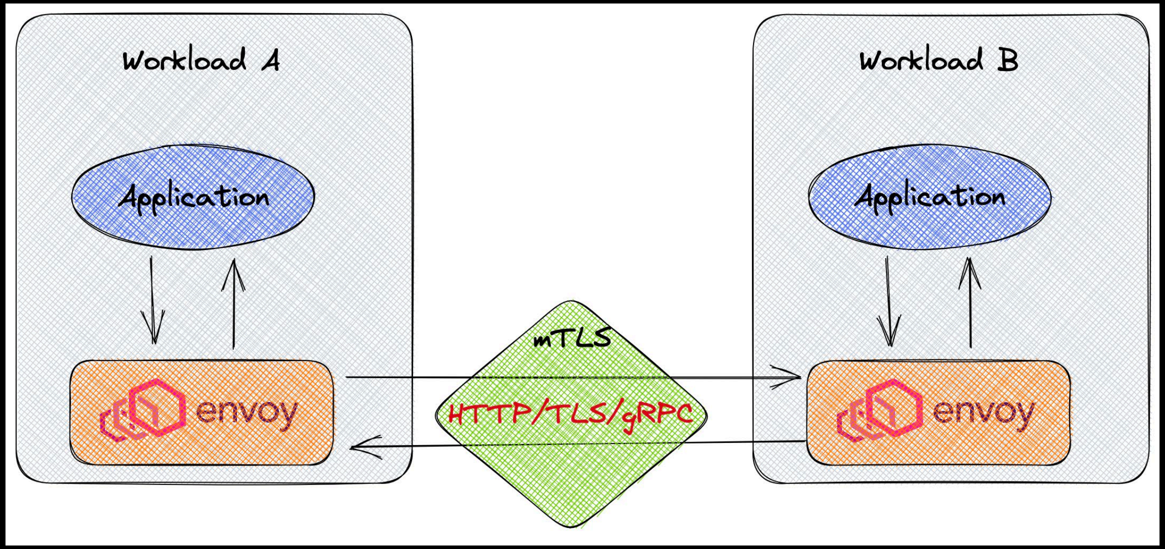


- Separate, isolated environment for code execution
- Guaranteed confidentiality/integrity for code/data
- Hardware-based isolation
- ARM TrustZone – Cortex-A and Cortex-M
- Secure boot chain & root of trust
- Communication between worlds – Secure Monitor & SMC

08 Trusted Platform Modules



- Storage area, crypto processor, security functions
- Tamper-evident and resistant
- Integrated into system board, usually separate chip
- Secure storage of root of trust
- Firmware and bootloader integrity verification
- Measure system configuration and software
- Enforce device-level security policies and monitor health



09 Service Meshes && mTLS

- Provides additional assurance that inter-service network traffic is being authorized from both the client and server
- This is especially important for many IoT devices which do not follow ironclad login procedures
- Utilize a service mesh to enforce strict adherence to mTLS across your application suite, to diminish and outright eliminate your risk of various malicious attacks

10 Admission Controllers

- Cosign Policy Admission Controller can protect namespaces in your kubernetes cluster by ensuring any scheduled OCI image was signed using a known key
- The policy controller can also perform policy-as-code verifications on attested payloads using Rego or Cue
- Additional admission controllers such as Gatekeeper or Kyverno can be used to enforce that workloads are only schedulable within policy controller protected namespaces

Demo

/Users/dmorrison/GolandProjects/github/2023-devsecops-days-securing-iot-session [15-gatekeeper]
% make deploy-chart
helm upgrade --install secure-iot ./deploy/charts/secure-iot/
Release "secure-iot" has been upgraded. Happy Helming!
NAME: secure-iot
LAST DEPLOYED: Wed May 10 19:04:42 2023
NAMESPACE: default
STATUS: deployed
REVISION: 6
NOTES:
1. Get the application URL by running these commands:
export POD_NAME=$(kubectl get pods --namespace secure -l "app.kubernetes.io/name=secure-iot-server,app.kubernetes.io/instance=secure-iot" -o jsonpath="{.items[0].metadata.name}")
export CONTAINER_PORT=$(kubectl get pod --namespace secure $POD_NAME -o jsonpath="{.spec.containers[0].ports[0].containerPort}")
echo "Visit http://127.0.0.1:8080 to use your application"
kubectl --namespace secure port-forward $POD_NAME 8080:$CONTAINER_PORT
/Users/dmorrison/GolandProjects/github/2023-devsecops-days-securing-iot-session [15-gatekeeper]
% c
SIGNED AND ATTESTED WITH CVE-FREE SBOM

Any Questions?

(RAFT)

Together, we'll make waves

www.goraft.tech