Security Policy – Framework

Security Policy (SP) driven Security Management (SM) and Orchestration for distributed Cloud Services

Dr. –Ing. Manfred Schäfer
Nokia Bell Labs, Security, Munich
Nokia Networks
Overview

- Motivation
  ‘Security Policies (SP)’ to support SM automation
- Architecture and Interworking of SM entities across Data Centers of different type
- Security Life Cycle Management
- SP Framework
  - Workflow Model
  - A Novel Domain Specific SP-Language (SP-DSL)
- SP-DSL Examples
- Outlook
Service/Provider Domains and Security Policies

Tenant/Service Domains may be built from resources offered by several providers (including different infrastructures)

- Each Tenant/Service has own Security Policies SP, which should be realized
  - Not a problem, if SPs are ‘within a single domain’

- Infrastructure Providers have own security expectations / SPs and offer security capabilities!

- Conflicts may arise, when crossing ‘domains’
  - In best case all domains belong to ONE organization ...
  - In the worst case SPs contradict to each other, which requires mitigation strategies.
  - In the general case, both, SPs/capabilities as well as related SM measures should be negotiated and agreed
Starting from ...

‘SM Gaps’ when analyzing recent ETSI NFV SEC proposals / earlier Security Orchestrator(s)

- Security Policies (SP) intensively used, but no framework defined
  - Not even requirements on SPs identified
  - No governance model regarding SP management and usage
- Multi-provider use cases / architecture missing so far
  - SM across ‘domain boundaries’ only addressed in so-called ‘single domain’ case (one organization view)
  - No negotiation, no conflict mitigation strategies
- Restricted to Telco Cloud (NFV type)
  - No approach supporting mixed services in a ‘mesh of clouds / DCs’
  - Two issues (SP, Multi-provider architecture) addressed in new work item (in IFA / SEC groups) .. Ongoing
- SP are know from other context ...
- As our primary goal, we aim to extend / enhance / integrate this, envisaging automated SM for distributed services
... towards ‘multiple-domains’, mixed services

Interfaces & functionality to enable SM cooperation
• Across administrative domains (with assigned SP authority)
• ‘cloud / DC boundaries’
• different cloud / DC types
• between Services / Infrastructure
• Centrally managed, but not excluding autonomous, local control

Main principles
• Negotiation (SM-interfaces, -services, SP <> Capabilities)
• Mediation (SPs, operation)
• Architecture for cooperative, SP driven automation of holistic, service-oriented SM

→ Security Management Service Points (SMSP)
Security Management Service Points (SMSP)

- SMSPs enable to provide and consume SM services
- ... are logical entities, which at 3 Layers implement functions to mediate and convert SM related SPs, constraints and services between different domains
- ... connecting SM entities in hierarchical manner, in at least 2 variants:
  - Cloud SM / Cloud SM entity (‘services’) or
  - Cloud SM / infrastructure SM
    -- internal or external infrastructure --
- ... allow ‘coupling’ global with local SM
  - control or delegate SM tasks (SPs) via/to SMSPs in lower hierarchy levels
  - Top down (steer) and bottom up (feedback)
  - At top level SM-root may correspond to the ‘security authority’ of a business model

"Need access to crypto unit CU"
"Take this * credential, enabling CU-user role"
Security Lifecycle Management Process (for services)

Scope of security policy (SP) based tasks

- Aggregate, prioritize and describe service SPs
- Pre-validation: high level conflict analysis

- Capability matching and conflict mitigation
- Agreements on SPs, interfacing, delegation and responsibility of involved SM entities

- SM topologies, architectures, tasks and interfaces
- SM service establishment and implementation (coordinated with service deployment)

- Detection of anomalies and deviations from intended SP scope (compliance analysis / validation, monitoring,..)
- SP or service based, reactive adaptations and operations
- Termination: Clean-up and protection of retained data
Use cases for SPs (‘virtualization specific’)

Fundamental Security Functions for distributed, virtualized Services
- Security domains and security zones
- Organization of SM entities, SM access control
- Link / data protection / SDN

Exploiting HW / platform rooted security for services
- Logical & physical isolation / separation of processes and traffic
- Trusted environments, crypto support / secure storage and processing

Security Mechanisms for VNF/VNFC (C=components)
- PKI integration / lifetime VNF/VNFC identities
- Group-key management and private key protection
- Insertion of (individual) security functions / controls

Monitoring use cases
- Exploiting platform/infrastructure capabilities, such as sensor placement, configuration and data retrieval
- Compliance validation / hardening
Automation through formalization -
Towards a novel domain specific SP language (SP-DSL)

Typically, existing SP languages are ..

Specifically tailored to dedicated security problems (like RBAC, XACML, GBP,..), hard to extend to cover the holistic, generic, and dynamic ‘SM for hybrid service’ area we are interested in

Design Goals

Domain specific SP Language shall ..

fit into the context of ‘orchestration systems’,
leverage SP/Service templates and catalogues
– modeling security needs from high-level to detailed

AND

.. meet RQs on ‘Expressiveness’
• Topology, location, grouping, segmentation / isolation, ...
• Time, counters, ..
• Conventional security capabilities (integrity, encryption, AAA,..)
• Conditional, built-in reactions (e.g., on events, compliance violation)
• Capture static as well as dynamic security requirements
• Delegation / permissions / interfaces (on SM)
• Extensibility (own or legacy SP elements, active behavior / code)
• ...

.. meet RQs on ‘automatable methods’
• SP Compositions / partitions
• SP Negotiation – RQ / Capability matching
• SP Transformation (SP \( \rightarrow \) deployment, runtime)
• Conflict and consistency analysis
• ...

9 26/09/2017 © Nokia 2017
Workflow Model

Domain Specific Language (DSL)

**DSL Templates**
- "SP – Catalogue"
- "Category – Catalog"
- "Base-Scenarios"

**Source Data Model (SDM) Templates**
Pre-existing SDM, e.g., TOSCA/Object Oriented (OO).

**‘Holistic’ DSL Model**
Specifying SPs in concrete scenarios

**DSL Processing**
E.g., SP – Analysis, Negotiation-Support, Partitioning

**‘Internal’ Parser / Translator**

**Extended ‘Target’ data models (TDM)**
associated with GROUPS, OBJTYPES / Relations, .., e.g., TOSCA / OO

**SM System**
Deployment, Runtime Workflows

**Deployment, Execution**

**Notes:**
- Source data model may be same as or different from target data model
SP Language (Categories, Assignments, Relations)

Generic Security Managed Objects (SMO)

Categories

Basic Object Categories

User
SM Entity
Service
Entity
Link

Generic, fundamental Security Requirement types (SRQ)

Security RQs / Policies

Platform
Authz
Conf
Integrity
Authn
Event

Individualized SMO types

= HSS
= 8
HSS_8

= PKI
= 12
PKI_12

Individualized SRQ types

= Key
= Data
= Pin
= encrypt
= in HW
= in SW
K_prot_PKI
D_prot_HSS

ASSIGNMENTS

Relations

Associate individual SRQ with individual SMO

ASSIGNMENTS
SP-DSL (Targeted Security Policies)

**Objects** can be taken from catalog and personalized for a specific scenario, e.g., ‘virtual PKI_CA’, ‘vMME’

‘**SRQ/Policy**’ can be taken from catalog and personalized for a specific scenario, e.g., ‘protect Signing Key’

SRQ-refinements / aggregations can be assigned as needed, e.g., ‘Use Hardware Security Module (HSM)’, FIPS compliance, ..

- Through **Security Relations**, Security Requirements / Polices (SRQ / SP) are assigned to SMOs, forming **Targeted Security Policies TSP**
- Security Relations include pre-conditions and post-conditions (constraints) → CCSR
SP-DSL (Groups, Types of Relations)

Through Relations SP
• may be assigned to (dynamic) groups of objects
• may be organized in groups themselves

... SP may be assigned
• to single objects / groups
• between objects / groups (mutual relation)

*) may be further refined
**SP-DSL (Reactive Rules)**

Event-condition/constraints-action rules (ECCA) (taking influence on the run-time system)

```plaintext
... OBJTYPES {
  ObjectCat A, B; // object category
  SRQCat SRQ; // SRQ category
  Relation RelName (RelCondition: SRQ -> (A,B): qualifier: RelConstraints);
  // Methods...
  // Rules ... like event-condition-action, for runtime control
  // pre-condition: (condition); post-condition: (constraints)
  Res EcAname On (EcEvent) if (EcCondition) then (EcAction, EcConstraints);
  // note: EcAction may demand enforcement of a certain Relation or SRQ
}
...
```

ECCA rules describe event-driven, policy controlled SM behavior in predictable situations.

---

**Exemplary Monitoring Life-cycle**

- **Platform B under attack**
- **Other platforms not affected**
- **Service scaled-in**
- **Service is distributed**
- **Clean VMs and move to 'other' platform**
- **Add vFW and adapt firewall rules across DC**
- **But only to those supporting an HSM**
- **Service must be kept running**

---

**Security related**

- **Business related**

---

**SP Protected Service**

- **Security Relevant Events**
- **SP controlled Security Modifications**
- **SP driven**
- **Monitoring Prioritized, gathered events**
- **Event Queue**
- **Handled by SP?**
- **Immediate reactions**
- **Conditional SPs**
- **External Compliance Logic**
  - Evaluation and determination of reactions
SP-DSL (Reactive Rules)

ECCA for Compliance Validation (taking influence on the run-time system)

OBJTYPES {...
  ObjectCat A; // object category
  SRQCat SRQ; // SRQ category
  Relation RelName (RelCondition: SRQ -> (A); qualifier: RelConstraints);
  // Methods...
  extern prglanguage */ e.g., script */ bool ValMethod(relation_reference, ValResT& cause);
  // relation_reference refers to a specified RelName
  // ValResT is a result value type, describing the reason (cause) for an invalid / valid proof of validation
  // & is a reference operator, as in C++, providing access to the 'cause' parameter
  // ValMethod is an external function, implemented in a programming language, as specified with planguage
  //
  // for compliance Validation
  Rest EccavalidationName
    On (EcTrigger) if (!ValMethod(RelName, cause)) then (EcAction(cause), EcConstraints(cause));
    // Trigger may be an event or a periodic action (like 'every(day, 12:00)')
    // note: EcAction may demand enforcement of a certain Relation or SRQ
    // note: EcAction and EcConstraints may also be external functions
...}

Example: e.g., PKI/CA
e.g., Issue CaA certificate ...

every month – check certificate – cert-renewal; respecting CERTIFICATE POLICY
**SP-DSL (Example)**

**BasicSecService**
Showing use of relations and associations in a simplified service scenario

**ASSIGMENTS**
- BOC_ENTITY VSF ("VM", "VNF", "VSF", ...);
- BOC_SMENTITY VSMF ("VM", "VNF", "VSMF", ...);
- BOC_ENTITY PF("vHW", "PFtype", ...);
- BOC_LINK VSFVSMFLink(...);
- SRQ_PFCat SecureBoot(...);
- SRQ_INSTCat SWISol(...);
- SRQ_INSTCat SWIsolate("physical separation", ...);
- SRQ_CONF Cat ConfProtLink(...);

**GROUPS**
// not used here

**OBJTYPES**
- SmallSecService
  - VSF mVSF;
  - VSMF mVSMF;
  - VSFVSMFLink mVSFVSMFLink;
  - PF mPF1, mPF2;
  // derived from SRQs
  - SWIPInst mSWIPInst;
  - SWIsol mSWIsol;
  - ConfProtLink mConfProtLink;
  - SecureBoot mSecureBoot;
  // Relations
  - Relation r1 : mSWIsol (\(\Rightarrow\)) (mVSF, mVSMF) : mandatory);
  - Relation r2 : mSWIPInst -> (mVSF, mVSMF) : mandatory);
  - Relation r3 : mConfProtLink -> mVSFVSMFLink : mandatory);
  - Association (connect /*reason, purpose */: mVSFVSMFLink \(\Leftrightarrow\) mVSF & mVSMF);
  // variant A
  - Relation r4A (:mSecureBoot (-\(\Rightarrow\)) (mVSMF, (mPF1, mPF2)) : mandatory);
  // variant B
  - Relation r4B (:x=mSecureBoot (-\(\Rightarrow\)) (mVSMF, (mPF1, mPF2)) : deploy (mVSMF, x));
  // variant C
  - Relation r4C1 (if (intended (mPF1, mVSMF)): mSecureBoot -> mPF1: mandatory);
  - Relation r4C2 (if (intended (mPF2, mVSMF)): mSecureBoot -> mPF2: mandatory);

**VALIDIF(r4A \(\land\) r4B \(\land\) (r4C1 & r2C2)): // alternative relations;
// \(\land\) = XOR**
SP-DSL (Example, extending SM, ‘service’)

Basic-SecO-managed-Service
Outlook

- **SP-DSL Refinements and Evaluation**
  - Modeling selected use cases, yet more formalization (e.g., on ‘conditions’, ‘functions’, policy import, ...)
  - Catalogue elements (BOC, SRQ)
- **Examination of appropriate Target Data Models / APIs**
  - e.g., TOSCA, YAML, YANG, HOT, GBP ...
- **Negotiation and partitioning concepts**
- **Analysis / logic reasoning**
  - Identification of conflicts and mitigation e.g., by (partly) mapping DSL models into Prolog
- **Ongoing Development and Demonstrator:** PLANETS ‘Security’, Master Thesis

**PLANETS:**  http://www.sendate.eu/de/sendate-planets-de

**@Nokia:**  michael.montag@nokia-bell-labs.com, manfred.schaefer@nokia-bell-labs.com
  fakhraddin.panakhov.ext@nokia.com (master student)
Backup Slides
Use cases for SPs (‘virtualization specific’)

• Fundamental Security Functions for distributed, virtualized Services
  • Security domains and security zones
  • Organization of SM entities, SM access control
  • Confidentiality and integrity of links and data, etc.

• Exploiting HW / platform rooted security for services
  • Logical & physical isolation / separation of processes and traffic
  • Trusted environments, crypto support / secure storage and processing

• Security orchestrated Key-Management for VNF/VFS, integrating PKI

• Security Mechanisms for VNFC (C=components)
  • Group-key management and private key protection
  • Insertion of (individual) security functions / controls

• Static/dynamic FW/VPN/.. requirements in distributed service

• Monitoring use cases
  • Exploiting platform/infrastructure capabilities, such as sensor placement, configuration and data retrieval
  • Compliance validation / hardening

• SDN scenarios
Automation through formalization -
Towards a novel domain specific SP language (SP-DSL)

Typically, existing SP languages are ..
• .. mostly focused on *specific* security realms, such as ‘RBAC’, ‘XACML’, firewall configurations (‘Group Based Policies’),..
• .. difficult to adapt or extend (e.g., databases underneath, tailored to mathematical model with a well defined scope)
• .. not designed to cover the holistic, generic and dynamic ‘SM for hybrid service’ area we are interested in

Design Goals
Domain specific SP Language shall ..
• .. be tailored to the ‘SM for service’ scope (preferring declarative approach, with separate analysis modules)
• .. fit into the context of ‘orchestration’ systems
• .. be capable to integrate external functionality (e.g., scripts or library modules or even other SP language constructs)
• .. leveraging SP/Service templates and catalogues -- from high-level to detailed

.. meet RQs on ‘Expressiveness’
• Topology, location, grouping, segmentation / isolation,...
• Time, counters, ..
• Conventional security capabilities (integrity, encryption, AAA,..)
• Conditional, built-in reactions (e.g., on events, compliance violation)
• Capture static as well as dynamic security requirements
• Delegation / permissions / interfaces (on SM)
• Extensibility (own or legacy SP elements, active behavior / code)
• ...

.. meet RQs on ,automatable methods‘
• SP Compositions / partitions
• SP Negotiation – RQ / Capability matching
• SP Transformation (SP → deployment, runtime)
• Conflict and consistency analysis
• ...

21 26/09/2017 © Nokia 2017
SP-DSL (Targeted Security Policies)

OBJTYPES {
    ObjectCat      A, B;
    SRQCat         SRQ1, SRQ-11;

    Relation RelName (RelCondition: SRQ-1 -> (A, B); qualifier: RelConstraints);
    // pre-condition: (condition); post-condition: (constraints)
}

Pre-condition
... IF HSS is instantiated in Germany ...

Qualifier
... Optional, mandatory, ...

Post-condition (after deployment, only if ...)
... Audited once a year
... Certificate Policy (CP) #2348 to be applied
... other applicable policies could be mentioned

(may include things that require human control)
The **IMPORT** attribute enables to integrate existing SP frameworks, e.g., for Access Control. Parameters / templates are used, which are assigned via an Assignment and / or Relation (may be combined with USING attribute).

**ASSIGNMENTS**

```
ASSIGNMENTS {
  ...
  SRQ_AUTHZCat
  SRQauthz_ACR_RBAC IMPORT RBAC_ACPL ( // example: interface, mapping to an existing RBAC framework
    RBAC_ACPL_policies ( // parameters, as required for ACPL
      PAllowConfig_wr
        entities_users <*> 'source' template, filled with
        node/user requiring access */>,
        roles (Admin,...),
        permissions (read,write: config-files),
        target <*> 'target' template, filled with
        node for which access shall
        be controlled*/>;
    );
    PAllowOther entities_users ... ( ... )
  ... 
  ... }
```

**OBJTYPES**

```
OBJTYPES {
    object_ACExample {
      BOC_ENTITY mVNFA (...), mVNFB (...);
      BOC_SMENTITY mVNFB_RBAC_SVR (...),
      SRQauthz_ACR_RBAC mSRQauthz_ACR_RBAC;
    }
    Relation RBACRule
      (:mSRQauthz_ACR_RBAC <mVNFA, mVNFB> ->) (mVNFA, mVNFB)::)
    USING mSRQauthz_ACR_RBAC;
  }
```

**Diagram**

- **mVNFA** (source)
- **mVNFB** (target)
- **mVNFB_RBAC_SVR**
- **RBAC_ACPL**
- **SRQauthz_ACR_RBAC**

**ACPL:** hypothetical policy language for Access Control
Example SP-DSL
Example SP-DSL

ASSIGNMENTS {
  COMBINE BOC_ENTITY, BOC_INSTANCE, BOC_IMAGE COC_VSNF; // definition of combined category
  COC_VSNF // its usage
  VFPNGW ("VSNF","VPNGW",... /*from BOC_INSTANCE , BOC_IMAGE*/ "imgfile"),imgVPNGWhash,...),
  HSS ("VNF","HSS",... /*from BOC_INSTANCE, BOC_IMAGE*/..., "imgfile"),imgHSSHash,...),
  vFW ("VSNF","VFW",... /*from BOC_INSTANCE, BOC_IMAGE*/..., "imgfile"),imgvFWhash,...),
  MME ("VNF","MME",... / other elements ...
  COMBINE BOC_SMENTITY, BOC_INSTANCE, BOC_IMAGE COC_SMVNF;
  COC_SMVNF
  SEM ("VSMF","SEM",...),
  SecO ("VSMF","SecO",...);
  BOC_ENTITY
  Platform ("PF"),...
  SOPF ("SecOPlatform",...);
  BOC_INSTANCE
  PhFW ("HW","FW",...);
  // security requirements
  SRQ_PFCat SBHW ("HW supp. SecureBoot",...); // platform shall be supported using HW like TPM & infrastructure
  SRQ_INSTCat SecOpHSep ("Not on service platform",...); // platform used for service instances
  SRQ_FILTERCat Filt6a ("FilterTrafficDPI", "ApplyFilterOnPlatform", (from/to) MME, (from/to) HSS, SCTP/IP,
  "default deny",...);
  SRQ_INTGRCat SRQSWPStaticA6Load ("SWP w. Proof of Origin",...); // signed images based on digital signature & PKI
  SRQ_CONFCat SRQVPNAcrossDC ("Connection over VPN",... ) USING VFPNGW;
  BOC_LINK
  6aPath ("path", "S6a-traffic", "EPS", "IP", "S6a-interface",...);
  SRQ_LINK PATHProtect ("Protect path", "encryption", "filtering", "authentication",...); // high-level SRQ, intents
  SRQ_PFCat PhFwInFrontPerDC ("Physical FW in Front of all platforms", "applicable per DC",...);
}

GROUPS {
  COC_VSNF GrpVSNF (VPNGW=0..2n=2, HSS, VFW, MME);
  // images are implicitly in this group;
  // VPNGW is a list composition of (pairwise) 0..2 elements
  COC_SMVNF GrpSMVNF (VSMF, SecO); // images are implicitly in this group
}

OBJTYPES {
  // through OBJTYPES referring to each other, we aim at modeling of administrative domains
  OPhysicalNWFunctions ( // physical domains
    mLCPFW=1..2 (PhFW); // List composition " at least one physical firewall"
  )
  OPlatforms ( // Infrastructure domains
    OPhysicalNWFunctions mPhysicalNWFunctions;
    mLCPlatforms<1..> (Platform); // LC " at least one Platform", considering all DCs together
    mLCSOPF<1..> (SOPF) ; // LC " at least one SOPF", considering all DCs together
    SBHW mSBHW;
    PhFwInFrontPerDC mPhFwInFrontPerDC;
    Relation ROP1 (mSBHW -> mLCSOPF: mandatory); // platform used for service instances
    Relation ROP2 (mVPNAcrossDC -> (mLCSOPF, mLCPFW): mandatory)
  )
  USING mOPhysicalNWFunctions.mLCPFW;
  OService ( // service domain
    OPhysicalNWFunctions mOPhysicalNWFunctions;
    mGrpVSNF (GrpVSNF);
    mGrpSMVNF (GrpSMVNF);
    ...
  )
  // define all SRQ members here:
  mS6apath mS6apath A6a ("connect":this/*mS6apath*/ <> mLCPFW);
  mS6apath mLCPFW;
  mOPlatforms mLCPFW;
  // as VPNGW is a LC, too, we probably could simplify this, writing
  Relation RO1 (mSecOpHSep -> mLCSOPF: mandatory);
  Relation RO2 (mVPNAcrossDC -> mS6apath: mandatory);
  Relation RO3 (mPATHProtect -> mS6apath: mandatory);
  Relation RO4 (mPATHProtect -> mS6apath: mandatory);
  Relation RO51 (mVPNAcrossDC -> mS6apath: mandatory);
  Relation RO52 (mVPNAcrossDC -> mS6apath: mandatory);
  Relation RO53 (mPATHProtect -> mLCSOPF: mandatory);
  Relation RO54 (mS6apath: mandatory);
  Relation RO55 (mS6apath: mandatory);
  Relation RO56 (mS6apath: mandatory);
  USING (mGrpVSNF.vFW, mOPhysicalNWFunctions.mLCPFW);
}
http://www.sendate.eu/de/ueber-das-projekt/
„Ziele und Vorgehen


Die im Projekt untersuchten Technologien bergen durch ihren komplexen Aufbau Angriffsmöglichkeiten für neuartige Cyber-Attacken. Um die IT-Sicherheit zu verbessern, werden daher auch Schutzmechanismen erforscht. Für eine sichere und zuverlässige Funktion der Netze ist dabei entscheidend, dass die Schutzmechanismen gut aufeinander abgestimmt und automatisiert werden können.

Innovationen und Perspektiven

Die Projektergebnisse fließen unter anderem in Konzepte und Produkte für die nächste Generation des mobilen Internets (5G) ein. Im Projekt erforschte Hardwaresicherheitsmodule und Verschlüsselungsbausteine bilden die Grundlage für Cloud-Infrastrukturen, die Daten in der Nähe des Nutzers sicher speichern und verarbeiten können. Damit ermöglichen die Projektergebnisse neue Anwendungen und Dienste in der Industrie 4.0: Geringe Reaktionszeiten erlauben sogar die Steuerung zeitkritischer Prozesse in der Produktion aus der Cloud. Sensible Geschäftsdaten sind dabei besser geschützt, da sie nicht über Länder mit weniger strengen Datenschutzregeln fließen. So können insbesondere kleine und mittelständische Unternehmen, die zumeist über keine eigene leistungsfähige IT-Infrastruktur verfügen, die Vorteile der Cloud voll ausnutzen.“