

Security Pattern Catalog Schema: Common Structure.

(schema/SecurityPatternCatalogNaiveSchema.owl)

Andrei Brazhuk (brazhuk@grsu.by)

<u>Property</u>	<u>Range and predefined values</u>	<u>Description</u>	<u>Example</u>
Class: Pattern <u>Class: SecurityPattern</u> (Defined class) <u>Class: ThreatPattern</u> (Defined class) <u>Class: MisusePattern</u> (Defined class)			<i>pattern_SecureDistributedPublishSubscribe IoT</i>
<u>Metadata:</u> * To describe a pattern you should use the “textIntent”, “textProblem”, “textSolution*”, “textConsequences”, “textImplementation” fields. And in some cases it is impossible to put a full description of the pattern because of copyright and trademark, so the "textreview*" fields retell a content of the pattern in own words. In the second case links to pattern’s original document should be provided.			
textName	xsd:String	Pattern’s primary name	<i>Secure Distributed Publish/Subscribe (P/S) pattern for IoT</i>
textAKAName	xsd:String	Pattern’s alternative name(s)	
textAuthor	xsd:String	Pattern’s author(s)	<i>Eduardo B. Fernandez Nobukazu Yoshioka Hironori Washizaki</i>
textURL	xsd:String	URL(s) of webpage that describes a pattern	https://www.researchgate.net/publication/339103887_Secure_Distributed_PublishSubscribe_PS_pattern_for_IoT
textPDF	xsd:String	Downloadable URL(s) of pattern’s PDF	
textReference	xsd:String	A bibliographic description of a primary paper, describing a pattern	<i>E.B.Fernandez, N. Yoshioka, H. Washizaki, “Secure Distributed Publish/Subscribe (P/S) for IoT, 2020. Procs. Asian PLoP’20, March 4 6, Taipei, Taiwan. 9 pages.</i>
textReviewContext	xsd:String	A brief text description of pattern’s context	Something like that: <i>Information exchange between IoT/IIoT devices (e.g. smart thermostats or sprinkler systems, different sensors) with minimal security control and cloud/fog applications.</i>

textReviewProblem	xsd:String	A brief text description of pattern's problem	Something like that: <i>Subscribers (S) register and receive messages of their interest sent by a publisher (P). The main concerns are how to organize the interactions between them securely, avoiding rogue participants, insecure communications, unwanted P/S operations.</i>
textReviewSolution	xsd:String	A brief text description of pattern's solution	<i>In addition of the standard P/S functions it is possible to use secure channels for protected communications, access control for restricting the actions of publishers and subscribers, security logging, and digital signatures.</i>
textIntent	xsd:String	Pattern's Intent (full text)	<i>In an IoT system, decouple the publishers of events from those interested in the events (subscribers). Subscription and publication are performed securely.</i>
textContext	xsd:String	Pattern's Context (full text)	
textProblem	xsd:String	Pattern's Problem (full text). It includes forces that constrain the solution.	
textSolution	xsd:String	Pattern's Solution description.	
textSolutionStructure	xsd:String	It describes the structure (static view) of the solution and some dynamic aspects in the form of sequence diagrams for a use case.	
imgSolutionStructure	xsd:String	Downloadable URL(s) of a solution structure diagram(s).	
textSolutionDynamics	xsd:String	It describes the dynamic aspects of the solution with diagrams.	
imgSolutionDynamics	xsd:String	Downloadable URL(s) of a solution dynamics diagram(s).	
textImplementation	xsd:String	The objective of this section is to describe what one should consider when implementing the pattern. This can be a set of general recommendations, or a sequence	

		of what to do to use the pattern. It may include some sample code, if appropriate.	
textConsequences	xsd:String	This section indicates the BENEFITS and LIABILITIES of the solution embodied in this pattern. The benefits should match the forces in the Problem section. Benefits that do not correspond to any force may appear.	
textKnownUses	xsd:String	To accept a solution as a pattern, it should be found at least three examples of its use in real systems.	

Organization and scope:

hasType	Type <u>Predefined items:</u> type_SecurityPattern type_ThreatPattern type_MisusePattern	Type of a pattern.	<i>hasType value type_SecurityPattern</i>
hasTemplate	Template <u>Predefined items:</u> template_POSA template_GOF	Template, used to describe a pattern. It can be POSA or GOF. POSA stands from “Pattern Oriented Software Architecture”. GOF stands from “Gang of Four”.	<i>hasTemplate value template_POSA</i>
hasGroup (inverse: isGroupOf)	Group	Tells to which group a pattern belongs to.	<i>hasGroup value patterngroup_SecureMiddleware</i>
usesPattern (inverse: isUsedBy)	Pattern	Enumerates patterns that are used by this one.	<i>pattern_RoleBasedAccessControl pattern_Authenticator pattern_SecurityLoggerAuditor pattern_SecureChannel</i>
relatesTo (symmetric)	Pattern	Enumerates patterns that are related to this one.	<i>pattern_SecurePS pattern_Broker</i>

			<i>pattern_SecureChannel</i> <i>pattern_EnterpriseServiceBus</i> <i>pattern_Authorizer</i> <i>pattern_IoTSegmentation</i>
isChildOf (inverse: isParentOf)	Pattern	For a concrete pattern shows from which abstract pattern it has been made. It can be possible to use the class assertion here, e.g. create a hierarchy with abstract patterns on the top and concrete ones at the bottom.	<i>IsChildOf</i> value <i>pattern_SecurePublishSubscrib</i>

Common characteristics:

hasDomain	Domain <u>Predefined items:</u> domain_FogComputing domain_EdgeComputing domain_InternetOfThings domain_SCADA domain_Military domain_ECommerce domain_GridComputing <u>Class:</u> CloudComputingDomain <i>domain_CloudComputing</i> domain_IaaS domain_PaaS domain_SaaS domain_NFV	Tells to which domain(s) a pattern belongs to. Domain is a large functional field of Information Technologies (IT), like Cloud Computing, Internet of Things. It might be less gigantic, like IaaS or NFV.	<i>hasDomain</i> value <i>domain_InternetOfThings</i>
hasArchitecturalLayer	ArchitecturalLayer <u>Predefined items:</u> <u>Class:</u> ApplicationArchitecturalLayer al_ClientLayer al_LogicLayer al_DataLayer <u>Class:</u> PlatformAndOperatingSystemLayer	Shows a common architectural domain, to which a pattern relates, like Applications, Platform and Operating systems, also Communications [Vale, 2019]. Instances are taken from [VanHilst, 2009]	<i>hasArchitecturalLayer</i> value <i>al_ClientLayer</i> <i>hasArchitecturalLayer</i> value <i>al_DistributionLayer</i>

	al_PlatformAndOperatingSystem <u>Class: CommunicationArchitecturalLayer</u> al_DistributionLayer al_TransportLayer al_NetworkLayer		
hasConstraintLevel	ConstraintLevel <u>Predefined instances:</u> cl_RegulatoryLevel cl_OrganizationalLevel cl_HumanLevel cl_MechanismLevel	<p>Refers to four levels of constraint: mechanism, human (operator or developer), organizational, and regulatory(Leveson, 2004).</p> <p>Instances are taken from [VanHilst, 2009]</p>	<i>hasConstrainLevel value</i> <i>cl_MechanismLevel</i>
hasResponseType	ResponseType <u>Predefined instances:</u> rt_Avoidance rt_Deterrence rt_Prevention rt_Detection rt_Mitigation rt_Recovery rt_Forensics	<p>“The response axis based on whether or not and attack happens and the extent, from not happening at all (avoidance), to completely happened and in the past (forensics).” [VanHilst, 2009]</p> <p>Instances are taken from [VanHilst, 2009]</p>	<i>hasResponseType value</i> <i>rt_Avoidance</i>
hasLifecycleStage	LifecycleStage <u>Predefined instances:</u> lc_ArchitectureAndDesign lc_BuildAndCompilation lc_Implementation lc_Installation lc_Operations lc_Requirements lc_SystemConfiguration lc_Deployment	<p>Tells which which system’s lifecycle stage a pattern is applicable.</p> <p>Frankly, most of the patterns are applicable on the Design (Architecture) stage, but it might be possible to have a few exceptions.</p> <p>Instances are taken from [CAPEC].</p>	<i>hasLifecycleStage value</i> <i>lc_ArchitectureAndDesign</i>
hasSecurityLevel	SecurityLevel <u>Predefined instances:</u> sl_PhysicalSecurity	<p>Tells to which field of security a pattern belongs to. To review.</p> <p>https://en.wikipedia.org/wiki/</p>	<i>hasSecurityLevel value</i> <i>sl_CommunicationAndDataSecurity</i>

	sl_PersonnelSecurity sl_CommunicationAndDataSecurity sl_OperationalSecurity	Physical_security https://en.wikipedia.org/wiki/Secure_communication https://en.wikipedia.org/wiki/Communications_security https://en.wikipedia.org/wiki/Operations_security	
--	--	--	--

Context characteristics:

hasAffectedFunction	Function see <i>schema_functions_components.pdf</i>	Tells which system function(s) a pattern affects.	<i>hasAffectedFunction value function_DistributeEventInformation</i> <i>hasAffectedFunction value function_DistributeSensorData</i>
hasAffectedComponent	Component see <i>schema_functions_components.pdf</i>	Tells which common component(s) a pattern affects.	<i>hasAffectedComponent value component_IoTApplication</i> <i>hasAffectedComponent value component_IIoTApplication</i> <i>hasAffectedComponent value component_CloudApplication</i> <i>hasAffectedComponent value component_FogApplication</i>

Security characteristics:

hasSecurityConcern	SecurityConcern <u>Predefined instances:</u> concern_AccessControl concern_Awareness	Tells which security concern(s) a pattern touches. “A security concern represents some security feature(s)” [Guan, 2016].	<i>hasSecurityConcern value concern_AccessControl</i> <i>hasSecurityConcern value concern_Audit</i>
---------------------------	--	--	--

	<p>concern_Training concern_Audit concern_Accountability concern_SecurityAssessment concern_Authorization concern_ConfigurationManagement concern_ContingencyPlanning concern_IdentificationAndAuthentication concern_IncidentResponse concern_Maintenance concern_MediaProtection concern_PhysicalProtection concern_EnvironmentalProtection concern_Planning concern_PersonnelSecurity concern_RiskAssessment concern_ServicesAcquisition concern_CommunicationsProtection concern_SystemProtection concern_SystemIntegrity concern_InformationIntegrity concern_ProgramManagement</p>	<p>Instances are taken from [NIST SP 800-53].</p>	<p><i>hasSecurityConcern value</i> <i>concern_InformationIntegrity</i></p> <p><i>hasSecurityConcern value</i> <i>concern_CommunicationsProtection</i></p>
<p>hasThreat (inverse: isThreatOf)</p>	<p>Threat</p> <p><u>Predefined instances:</u> see <i>schema_threats.pdf</i></p> <p><u>Class:</u> CommunicationsThreat threat_ManInTheMiddle threat_Interception threat_Flooding threat_ContentSpoofing threat_IdentitySpoofing threat_Footprinting (=threat_InformationGathering) threat_ProtocolAnalysis</p> <p><u>Class:</u> SoftwareThreat</p>	<p>Tells what threats a pattern describes with connection to component(s) and function(s), figured out on the previous stage.</p> <p>For security patterns defines the possible threats, met by a pattern. For attack pattern defines the possible threats, produced by an implementation of pattern.</p> <p>Instances are adopted from [CAPEC] (see <i>schema_threats.pdf</i>)</p>	<p><i>hasThreat value</i> <i>threat_PrivilegeEscalation</i></p> <p><i>hasThreat value threat_IdentitySpoofing</i></p> <p><i>hasThreat value threat_Interception</i></p> <p><i>hasThreat value threat_ContentSpoofing</i></p> <p><i>hasThreat value threat_Flooding</i></p>

	threat_SessionManipulation threat_AuthenticationBypass threat_PrivilegeEscalation threat_Excavation threat_CodeInjection threat_BufferManipulation threat_ExcessiveAllocation threat_ManipulationAPI threat_InputDataManipulation threat_EnvironmentManipulation threat_SharedDataManipulation threat_Malware		
--	--	--	--

Inferred characteristics:

hasPossibleAttack	Attack	Will be taken from [CAPEC] and other attacks' classifications.	
hasPossibleWeakness	Weakness	Will be taken from [CWE] and other weaknesses'/vulnerabilities' classifications.	

Class: Threat

Contains automatically assigned properties. Automatic reasoning procedures will get them from the internal data scheme.

hasThreatImpact (inverse: isThreatImpactOf)	ThreatImpact <u>Predefined instances:</u> see <i>schema_threats.pdf</i> ti_AlterExecutionLogic ti_BypassProtectionMechanism ti_ExecuteArbitraryCode ti_GainPrivileges ti_HideActivities ti_ModifyData ti_ReadData ti_ResourceConsumption ti_UnreliableExecution	Tells which negative impact(s) the threats, described by a pattern, have to component(s) and function(s). It obtains from the CAPEC attack descriptions (the hasThreat property here).	
--	---	--	--

hasSTRIDE (inverse: isSTRIDEof)	STRIDE <u>Predefined instances:</u> STRIDE_Spoofing STRIDE_Tampering STRIDE_Repudiation STRIDE_Information_Disclosure STRIDE_Denial_of_Service STRIDE_Elevation_of_Privilege	Tells which STRIDE item(s) a pattern touches. To do: map STRIDE & SO	
hasSecurityObjective (inverse: isSecurityObjectiveOf)	SecurityObjective <u>Predefined instances:</u> SO_AccessControl SO_Accountability SO_Authentication SO_Authorization SO_Availability SO_Confidentiality SO_Integrity SO_NonRepudiation	Tells which security objective(s) a pattern touches.	

References:

[Guan, 2016] H. Guan, H. Yang, and J. Wang, “An ontology-based approach to security pattern selection,” Int. J. Autom. Comput., vol. 13, pp. 168–182, Apr. 2016.

[Vale, 2019] A.P. Vale, E. B. Fernández, “An Ontology for Security Patterns”. Conference paper. 2019.

[VanHilst, 2009] VanHilst M. et al. A multi-dimensional classification for users of security patterns //Journal of Research and Practice in Information Technology. – 2009. – T. 41. – №. 2. – C. 87.

[CAPEC] <https://capec.mitre.org/>

[CWE] <https://cwe.mitre.org/>

[Fernandez, 2013] E. B. Fernandez, Security patterns in practice: designing secure architectures using software patterns. John Wiley and Sons, 2013.

[NIST SP 800-53] Security and privacy controls for federal information systems and organizations NIST Special Publication 800-43 revision 4 //NIST. – 2013.