Threats, threat impacts, security objectives, STRIDEs

(schema/SecurityPatternCatalogNaiveSchema.owl)
brazhuk@grsu.by

Here the CAPEC list of attack patterns (domain of Attack) is used. The CAPEC categories ("Software", "Communications") contain the pattens with the "Meta" abstraction. Most valuable (and most generic) of them have been chosen. The "ThreatImpact" and "SO" values have been added with some modifications.

Threat	ThreatImpact	SO	STRIDE	CAPEC	CAPEC references
Class: CommunicationsThreat					
Class: **ManInTheMiddleThreat** **threat_ManInTheMiddle**	ti_GainPrivileges ti_ModifyData ti_ReadData	SO_Integrity SO_Confidentiality SO_Authentication		CAPEC_94	CAPEC-94: Man in the Middle Attack **The attacker places himself in the communication channel between the two components** (typically client and server). **Whenever one component attempts to communicate with the other, the data first goes to the attacker, who has the opportunity to observe or alter it, and it is then passed on to the other component as if it was never observed.** This interposition is transparent **leaving the two compromised components unaware of the potential corruption or leakage of their communications** [CAPEC-94]. *Integrity – Modify data* *Confidentiality – Read data* *Confidentiality – Gain Privileges* *Access Control – Gain Privileges* *Authorization - Gain Privileges*
Class: **InterceptionThreat** **threat_Interception**	ti_ReadData	SO_Confidentiality		CAPEC_117	CAPEC-117: Interception **An adversary monitors data streams to or from the target for information gathering purposes.** This attack may be undertaken to solely gather sensitive information or to support a further attack against the target. This attack pattern can involve sniffing network traffic as well as other types of data streams (e.g. radio). [CAPEC-117] *Confidentiality – Read data*
Class: **FloodingThreat** **threat_Flooding**	ti_ResourceConsumption	SO_Availability		CAPEC_125	CAPEC-125: Flooding **An adversary consumes the resources of a target by rapidly engaging in a large number of interactions with the target**. This type of attack generally exposes a weakness in rate limiting or flow. When successful **this attack prevents legitimate users from accessing the service and can cause the target to crash.** [CAPEC-125] *Availability - Unreliable Execution* *Resource Consumption*
Class: **ContentSpoofingThreat** **threat_ContentSpoofing**	ti_ModifyData ti_ReadData	SO_Integrity SO_Confidentiality		CAPEC_148	CAPEC-148: Content Spoofing **An adversary modifies content to make it contain something other than what the original content producer intended while keeping**

					the apparent source of the content unchanged. Content can be modified at the source (e.g. modifying the source file for a web page) or in transit (e.g. **intercepting and modifying a message between the sender and recipient**). [CAPEC-148] *Integrity - Modify Data*
Class: **IdentitySpoofingThreat** **threat_IdentitySpoofing**	ti_GainPrivileges ti_ReadData	SO_Confidentiality SO_Authentication		CAPEC_151	CAPEC-151: Identity Spoofing **Identity Spoofing refers to the action of assuming (i.e., taking on) the identity of some other entity (human or non-human) and then using that identity to accomplish a goal**. An adversary may craft messages that appear to come from a different principle or use stolen / spoofed authentication credentials. **Alternatively, an adversary may intercept a message from a legitimate sender and attempt to make it look like the message comes from them without changing its content.** [CAPEC-151] *Confidentiality - Gain Privileges* *Integrity* *Authentication* *Access Control*
Class: **FootprintingThreat** **threat_Footprinting (=threat_InformationGathering)**	ti_ReadData	SO_Confidentiality		CAPEC_169	CAPEC-169: Footprinting **An adversary engages in probing and exploration activities to identify constituents and properties of the target.** Footprinting is a general term to describe a variety of information gathering techniques, often used by attackers in preparation for some attack. It consists of using tools to learn as much as possible about the composition, configuration, and security mechanisms of the targeted ~~application~~ system or network. [CAPEC-169] *Confidentiality – Read data*
Class: **ProtocolAnalysisThreat** **threat_ProtocolAnalysis**	ti_ReadData ti_ModifyData	SO_Confidentiality SO_Integrity		CAPEC_192	CAPEC-192: Protocol Analysis **An adversary engages in activities to decipher and/or decode protocol information for a network or application communication protocol used for transmitting information between interconnected nodes or systems on a packet-switched data network.** [CAPEC-192] *Confidentiality - Read Data* *Integrity - Modify Data*
Class: SoftwareThreat					
Class: **SessionManipulationThreat** **threat_SessionManipulation**	ti_GainPrivileges ti_ReadData ti_ModifyData	SO_Confidentiality SO_AccessControl SO_Authentication SO_Integrity		CAPEC_21	CAPEC-21: Exploitation of Trusted Credentials **Attacks on session IDs and resource IDs take advantage of the fact that some software accepts user input without verifying its authenticity.** Many server side processes are vulnerable to these

					attacks because the server to server communications have not been analyzed from a security perspective or the processes "trust" other systems because they are behind a firewall. **Session IDs may be guessed due to insufficient randomness, poor protection (passed in the clear), lack of integrity (unsigned), or improperly correlation with access control policy enforcement points.** [CAPEC-21] *Confidentiality - Gain Privileges* *Access Control* *Authorization* *Confidentiality - Read Data* *Integrity - Modify Data*
Class: **AuthenticationBypassThreat** **threat_AuthenticationBypass**	ti_GainPrivileges	SO_AccessControl SO_Authentication		CAPEC_114 CAPEC_115 CAPEC_112	CAPEC-114: Authentication Abuse CAPEC-115: Authentication Bypass **An attacker obtains unauthorized access to an application, service or device either through knowledge of the inherent weaknesses of an authentication mechanism, or by exploiting a flaw in the authentication scheme's implementation**. Authentication Abuse allows the attacker to be certified as a valid user through illegitimate means, while Authentication Bypass **allows the user to access protected material without ever being certified as an authenticated user**. [CAPEC-114, CAPEC-115] CAPEC-112: Brute Force In this attack, some asset (information, functionality, identity, etc.) is protected by a finite secret value. **The attacker attempts to gain access to this asset by using trial-and-error to exhaustively explore all the possible secret values in the hope of finding the secret (or a value that is functionally equivalent) that will unlock the asset.** Examples of secrets can include, but are not limited to, passwords, encryption keys, database lookup keys, and initial values to one-way functions. [CAPEC-112] *Confidentiality - Read Data* *Confidentiality - Gain Privileges* *Access Control* *Authorization*
Class: **PriviledgeEscalationThreat** **threat_PriviledgeEscalation**	ti_GainPrivileges	SO_AccessControl SO_Authorization		CAPEC_122 CAPEC_233	CAPEC-122: Privilege Abuse **An adversary is able to exploit features of the target that should be reserved for privileged users or administrators but are exposed to use by lower or non-privileged accounts.** Access to sensitive information and functionality must be controlled to ensure that only authorized users are able to access these resources. If access control mechanisms are absent or misconfigured, a user may be able to access resources that are intended only for higher level users. [CAPEC-122] CAPEC-233: Privilege Escalation **An adversary exploits a weakness enabling them to elevate their**

					privilege and perform an action that they are not supposed to be authorized to perform. [CAPEC-233]
Class: **ExcavationThreat** **threat_Excavation**	ti_ReadData	SO_Confidentiality		CAPEC_116	CAPEC-116: Excavation **An adversary actively probes the target in a manner that is designed to solicit information that could be leveraged for malicious purposes.** This is achieved by exploring the target via ordinary interactions for the purpose of gathering intelligence about the target, or by sending data that is syntactically invalid or non-standard in an attempt to produce a response that contains the desired data. As a result of these interactions, the adversary is able to obtain information from the target that aids the attacker in making inferences about its security, configuration, or potential vulnerabilities. Examplar exchanges with the target may trigger unhandled exceptions or verbose error messages that reveal information like stack traces, configuration information, path information, or database design. **This type of attack also includes the manipulation of query strings in a URI to produce invalid SQL queries, or by trying alternative path values in the hope that the server will return useful information.** [CAPEC-116] *Confidentiality - Read Data*
class: **CodeInjectionThreat** **threat_CodeInjection**	ti_ExecuteArbitraryCode	SO_Confidentiality SO_Integrity SO_Availability		CAPEC_242	CAPEC-242: Code Injection **An adversary exploits a weakness in input validation on the target to inject new code into that which is currently executing.** [CAPEC-242] *Confidentiality* *Integrity* *Availability*
Class: **BufferManipulationThreat** **threat_BufferManipulation**	ti_UnreliableExecution ti_ReadData ti_ModifyData	SO_Confidentiality SO_Integrity SO_Availability		CAPEC_123 CAPEC_129	CAPEC-123: Buffer Manipulation **An adversary manipulates an application's interaction with a buffer in an attempt to read or modify data they shouldn't have access to.** [CAPEC-123] *Availability - Unreliable Execution* *Confidentiality - Execute Unauthorized Commands* *Modify Data* *Read Data* CAPEC-129: Pointer Manipulation This attack pattern involves an adversary manipulating a pointer within a target application resulting in the application accessing an unintended memory location. This can result in the crashing of the application or, for certain pointer values, access to data that would not normally be possible or the execution of arbitrary code. Since pointers are simply integer variables, Integer Attacks may often be used in Pointer Attacks. [CAPEC-129]
class: **ExcessiveAllocationThreat**	ti_ResourceConsumption	SO_Availability		CAPEC_130	CAPEC-130: Excessive Allocation

threat_ExcessiveAllocation	ti_UnreliableExecution			CAPEC_131	**An adversary causes the target to allocate excessive resources to servicing the attackers' request, thereby reducing the resources available for legitimate services and degrading or denying services**. This attack uses one or a small number of requests that are carefully formatted to force the target to allocate excessive resources to service this request(s). [CAPEC-130] *Availability - Resource Consumption* CAPEC-131: Resource Leak Exposure An adversary utilizes a resource leak on the target to deplete the quantity of the resource available to service legitimate requests. Resource leaks most often come in the form of memory leaks where memory is allocated but never released after it has served its purpose. [CAPEC-131] *Availability - Resource Consumption* *Unreliable Execution*
class: **ManipulationAPIThreat** **threat_ManipulationAPI**	ti_UnreliableExecution ti_ExecuteArbitraryCode ti_ReadData ti_ModifyData	SO_Confidentiality SO_Integrity SO_Availability		CAPEC_113 CAPEC_586	CAPEC-113: API Manipulation **An adversary manipulates the use or processing of an Application Programming Interface (API) resulting in an adverse impact upon the security of the system implementing the API.** This can allow the adversary to execute functionality not intended by the API implementation, possibly compromising the system which integrates the API. API manipulation can take on a number of forms including forcing the unexpected use of an API, or the use of an API in an unintended way. [CAPEC-113] CAPEC-586: Object Injection **An adversary attempts to exploit an application by injecting additional, malicious content during its processing of serialized objects.** Developers leverage serialization in order to convert data or state into a static, binary format for saving to disk or transferring over a network. These objects are then deserialized when needed to recover the data/state. [CAPEC-586]
Class: **InputDataManipulationThreat** **threat_InputDataManipulation**	ti_UnreliableExecution ti_ExecuteArbitraryCode ti_ReadData ti_ModifyData	SO_Confidentiality SO_Integrity SO_Availability		CAPEC_153 CAPEC_137 CAPEC_248	CAPEC-153: Input Data Manipulation **An attacker exploits a weakness in input validation by controlling the format, structure, and composition of data to an input-processing interface.** By supplying input of a non-standard or unexpected form an attacker can adversely impact the security of the target. [CAPEC-153] CAPEC-137: Parameter Injection **An adversary manipulates the content of request parameters for the purpose of undermining the security of the target.** Some parameter encodings use text characters as separators. For example, parameters in a HTTP GET message are encoded as name-value pairs separated by an ampersand (&). If an attacker can supply text strings

					that are used to fill in these parameters, then they can inject special characters used in the encoding scheme to add or modify parameters. For example, if user input is fed directly into an HTTP GET request and the user provides the value "myInput&new_param=myValue", then the input parameter is set to myInput, but a new parameter (new_param) is also added with a value of myValue. [CAPEC-137] *Integrity - Modify Data* CAPEC-248: Command Injection **An adversary looking to execute a command of their choosing, injects new items into an existing command thus modifying interpretation away from what was intended.** Commands in this context are often standalone strings that are interpreted by a downstream component and cause specific responses. This type of attack is possible when untrusted values are used to build these command strings. Weaknesses in input validation or command construction can enable the attack and lead to successful exploitation. [CAPEC-248] *Confidentiality - Execute Unauthorized Commands* *Integrity* *Availability*
Class: **EnvironmentManipulationThreat**	ti_ReadData ti_ModifyData	SO_Confidentiality SO_Integrity			CAPEC-176: Configuration/Environment Manipulation **An attacker manipulates files or settings external to a target application which affect the behavior of that application.** For example, many applications use external configuration files and libraries - modification of these entities or otherwise affecting the application's ability to use them would constitute a configuration/environment manipulation attack. [CAPEC-176]
Class: **SharedDataManipulationThreat** **threat_SharedDataManipulation**	ti_ReadData ti_ModifyData	SO_Confidentiality SO_Integrity		CAPEC_124 CAPEC_26	CAPEC-124: Shared Data Manipulation **An adversary exploits a data structure shared between multiple applications or an application pool to affect application behavior.** Data may be shared between multiple applications or between multiple threads of a single application. Data sharing is usually accomplished through mutual access to a single memory location. If an adversary can manipulate this shared data (usually by co-opting one of the applications or threads) the other applications or threads using the shared data will often continue to trust the validity of the compromised shared data and use it in their calculations [CAPEC-124] CAPEC-26: Leveraging Race Conditions The adversary targets a race condition occurring when multiple processes access and manipulate the same resource concurrently, and the outcome of the execution depends on the particular order in which the access takes place. The adversary can leverage a race condition by "running the race", modifying the resource and modifying the normal

					execution flow. [CAPEC-26] *Confidentiality -Gain Privileges* *Access Control* *Authorization* *Integrity - Modify Data*
Class: **MalwareThreat** **threat_Malware**	ti_UnreliableExecution ti_ExecuteArbitraryCode ti_ReadData ti_ModifyData	SO_Confidentiality SO_Integrity SO_Availability			CAPEC-175: Code Inclusion **An adversary exploits a weakness on the target to force arbitrary code to be retrieved locally or from a remote location and executed.** This differs from code injection in that code injection involves the direct inclusion of code while code inclusion involves the addition or replacement of a reference to a code file, which is subsequently loaded by the target and used as part of the code of some application. [CAPEC-175] CAPEC-441: Malicious Logic Insertion **An adversary installs or adds malicious logic (also known as malware) into a seemingly benign component of a fielded system.** This logic is often hidden from the user of the system and works behind the scenes to achieve negative impacts.[CAPEC-441] *Authorization - Execute Unauthorized Commands* CAPEC-549: Local Execution of Code **An adversary installs and executes malicious code on the target system in an effort to achieve a negative technical impact.** Examples include rootkits, ransomware, spyware, adware, and others. [CAPEC-549] *Confidentiality - Execute Unauthorized Commands* *Integrity* *Availability*

[CAPEC] https://capec.mitre.org/
[CWE] https://cwe.mitre.org/

To do, to research, to modify

ThreatImpact	TechnicalImpact	CAPEC Impact	SO	STRIDE
ti_AlterExecutionLogic				
ti_BypassProtectionMechanism				
ti_ExecuteArbitraryCode		Execute Unauthorized Commands	SO_Authorization	
ti_GainPrivileges		Gain Privileges	SO_Confidentiality	

			SO_Authorization SO_Authentication	
ti_HideActivities				
ti_ModifyData		Modify data	SO_Integrity	
ti_ReadData		Read data	SO_Confidentiality	
ti_ResourceConsumption		Resource Consumption	SO_Availability	
ti_UnreliableExecution		Unreliable Execution	SO_Availability	

Original CAPEC
<Impact>Alter Execution Logic</Impact>
<Impact>Bypass Protection Mechanism</Impact>
<Impact>Execute Unauthorized Commands</Impact>
<Impact>Gain Privileges</Impact>
<Impact>Hide Activities</Impact>
<Impact>Modify Data</Impact>
<Impact>Other</Impact>
<Impact>Read Data</Impact>
<Impact>Resource Consumption</Impact>
<Impact>Unreliable Execution</Impact>

Original CWE
<Impact>Alter Execution Logic</Impact>
<Impact>Bypass Protection Mechanism</Impact>
<Impact>DoS: Amplification</Impact>
<Impact>DoS: Crash, Exit, or Restart</Impact>
<Impact>DoS: Instability</Impact>
<Impact>DoS: Resource Consumption (CPU)</Impact>
<Impact>DoS: Resource Consumption (Memory)</Impact>
<Impact>DoS: Resource Consumption (Other)</Impact>
<Impact>Execute Unauthorized Code or Commands</Impact>
<Impact>Gain Privileges or Assume Identity</Impact>
<Impact>Hide Activities</Impact>
<Impact>Modify Application Data</Impact>
<Impact>Modify Files or Directories</Impact>
<Impact>Modify Memory</Impact>
<Impact>Other</Impact>

<Impact>Quality Degradation</Impact>
<Impact>Read Application Data</Impact>
<Impact>Read Files or Directories</Impact>
<Impact>Read Memory</Impact>
<Impact>Reduce Maintainability</Impact>
<Impact>Reduce Performance</Impact>
<Impact>Reduce Reliability</Impact>
<Impact>Unexpected State</Impact>
<Impact>Varies by Context</Impact>