

OWASP TOP 10 IN AN INDUSTRIAL CONTEXT

György Kálmán, Ph.D.
mnemonic/NTNU/UiO
gyorgy.kalman@its.uio.no

mnemonic

Agenda



-
- Connected systems
 - Message
 - OWASP Top 10 (2017 Release Candidate)
 - A1 Injection
 - A2 Broken authentication
 - A3 Cross-Side Scripting
 - A4 Broken access control, back from 2004
 - A5 Security Misconfiguration
 - A6 Sensitive data exposure
 - A7 Insufficient Attack Protection (new)
 - A8 Cross-Site Request Forgery
 - A9 Using Components with Known Vulnerabilities
 - A10 Underprotected APIs (new)
 - Conclusion

| Connected Systems

Attack surface and security baseline

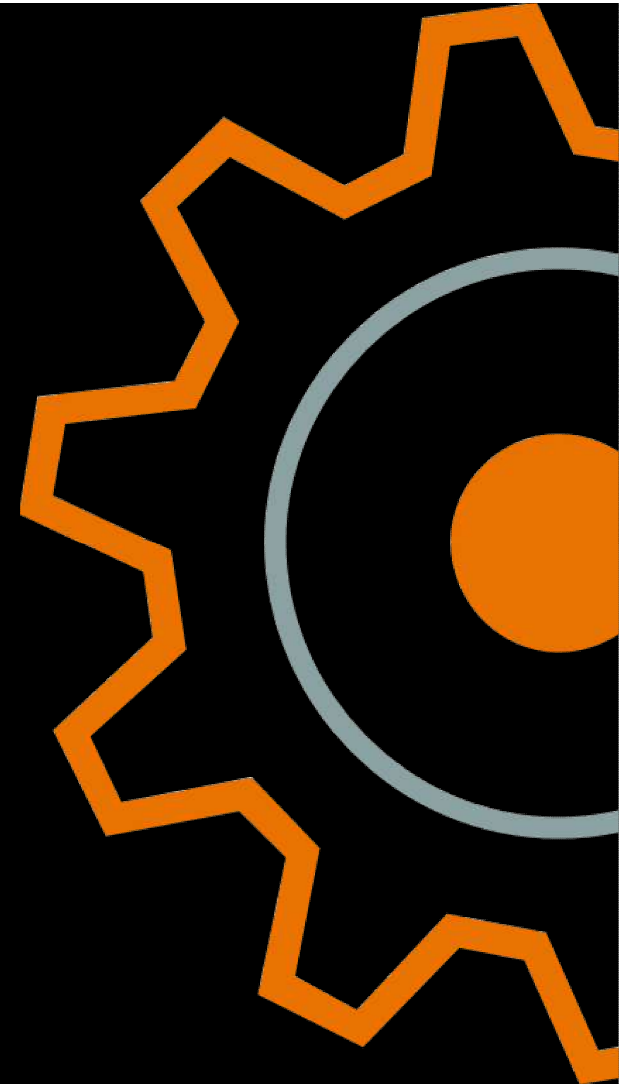
- Components
 - Automation
 - Infrastructure
 - Sensors
 - Customer equipment
- Connected plant and connected user services
- The attack surface extends dramatically with IIoT
 - Simple devices, resource constraints
 - Connected through third party networks
 - Configuration, management and life-cycle
 - Implicit trust between components, physical interfaces
- In the same time, getting closer to the customer
 - Devices merged into customer premises infrastructure
 - Security baselining on same level as customer services



| OWASP Top 10

Web application security

- A1 Injection
- A2 Broken Authentication and Session Management
- A3 Cross-Site Scripting
- A4 Broken Access Control
- A5 Security Misconfiguration
- A6 Sensitive Data Exposure
- A7 Insufficient Attack Protection
- A8 Cross-Site Request Forgery
- A9 Using Components with Known Vulnerabilities
- A10 Underprotected APIs



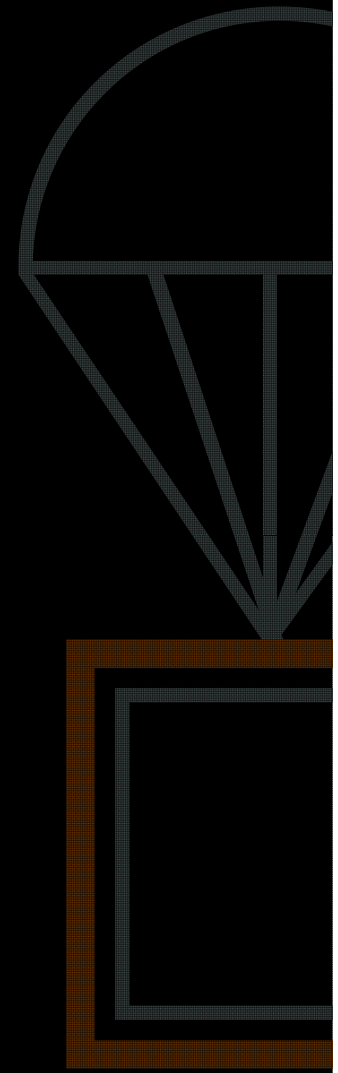
OWASP Top 10 IoT

A bit different

- I1 Insecure Web Interface
- I2 Insufficient Authentication/Authorization
- I3 Insecure Network Services
- I4 Lack of Transport Encryption
- I5 Privacy Concerns
- I6 Insecure Cloud Interface
- I7 Insecure Mobile Interface
- I8 Insufficient Security Configurability
- I9 Insecure Software/Firmware
- I10 Poor Physical Security

| Message

-
- OWASP Top 10 is a list of the most risky web app vulnerabilities
 - Test the devices and services against OWASP Top 10 to establish a common baseline
 - Low resources in the devices are not an excuse for not showing due care in security
 - OWASP Top 10 IoT is more specialised – maybe less available
-
- The references on vulnerabilities are randomly selected and there is no relation to how secure or insecure the vendor is



OWASP Top 10 – A1 Injection

- Example: Siemens WinCC login screen, SQL injection vulnerability, CVE-2013-3957
- Allows remote attackers to execute arbitrary SQL commands






- All figures from OWASP Top 10, 2017 Release Candidate
<https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top10-%20-%202017%20RC1-English.pdf>

Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider anyone who can send untrusted data to the system, including external users, business partners, other systems, internal users, and administrators.	Attackers send simple text-based attacks that exploit the syntax of the targeted interpreter. Almost any source of data can be an injection vector, including internal sources.	<u>Injection flaws</u> occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, XPath, or NoSQL queries; OS commands; XML parsers, SMTP Headers, expression languages, etc. Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws.		Injection can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover.	Consider the business value of the affected data and the platform running the interpreter. All data could be stolen, modified, or deleted. Could your reputation be harmed?



OWASP Top 10 – A2 Broken authentication

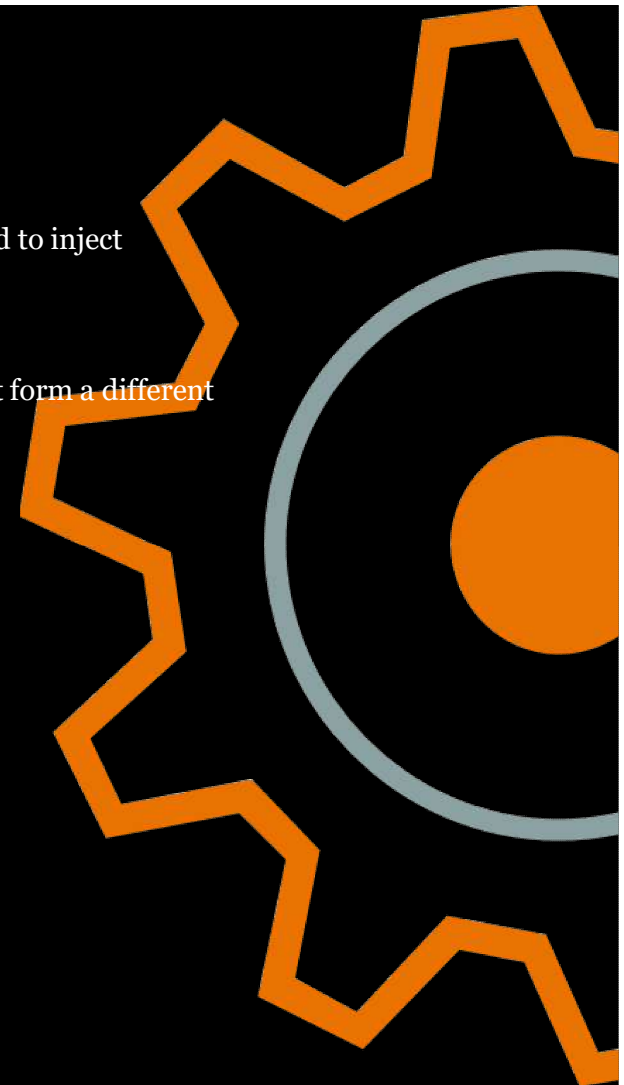
- Moxa: user/administrative level of access can easily circumvented, granting write access to user level. ICS-CERT advisory, ICSA-15-246-03
- Multiple vulnerabilities:
- Improper privilege management (this vulnerability)
- Resource exhaustion: crafted packet sent to the embedded browser causes the units to restart
- Cross-Site Scripting: An input field of the administrative web interface lacks input validation, which could be abused to inject JavaScript code.
- Default usernames and passwords are also a broken authentication vulnerability

					
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider anonymous external attackers, as well as authorized users, who may attempt to steal accounts from others. Also consider insiders wanting to disguise their actions.	Attackers use leaks or flaws in the authentication or session management functions (e.g., exposed accounts, passwords, session IDs) to temporarily or permanently impersonate users.	Developers frequently build custom authentication and session management schemes, but building these correctly is hard. As a result, these custom schemes frequently have flaws in areas such as logout, create account, change password, forgot password, timeouts, remember me, secret question, account update, etc. Finding such flaws can sometimes be difficult, as each implementation is unique.		Such flaws may allow some or even <u>all</u> accounts to be attacked. Once successful, the attacker can do anything the victim could do. Privileged accounts are frequently targeted.	Consider the business value of the affected data and application functions. Also consider the business impact of public exposure of the vulnerability.

OWASP Top 10 – A3 Cross-Side Scripting

- The previous example had several vulnerabilities, amongst others, also Cross-Side Scripting:
 - An input field of the administrative web interface lacks input validation, which could be abused to inject JavaScript code.
-
- I've met an example, where the web UI itself was implemented by the vendor to load javascript form a different domain, but also not protected against XSS






Threat Agents	Attack Vectors	Security Weakness	Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence VERY WIDESPREAD	Detectability AVERAGE	Impact MODERATE
Consider anyone who can send untrusted data to the system, including external users, business partners, other systems, internal users, and administrators.	Attackers send text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.	XSS flaws occur when an application updates a web page with attacker controlled data without properly escaping that content or using a safe JavaScript API. There are two primary categories of XSS flaws: (1) Stored , and (2) Reflected , and each of these can occur on (a) the Server or (b) on the Client . Detection of most Server XSS flaws is fairly easy via testing or code analysis. Client XSS can be very difficult to identify.	Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc.	Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability.



OWASP Top 10 – A4 Broken access control

- EWON routers and gateways, ICS-CERT ICSA-15-351-03

- The software allows an unauthenticated user to gather information and status of I/O servers through the use of a forged URL.
- eWON firmware web server allows the use of the HTML command GET in place of POST. GET is less secure because data that are sent are part of the URL.

					
Application Specific	Exploitability EASY	Prevalence WIDESPREAD	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider the types of authorized users of your system. Are users restricted to certain functions and data? Are unauthenticated users allowed access to any functionality or data?	Attackers, who are authorized users, simply change a parameter value to another resource they aren't authorized for. Is access to this functionality or data granted?	For data, applications and APIs frequently use the actual name or key of an object when generating web pages. For functions, URLs and function names are frequently easy to guess. Applications and APIs don't always verify the user is authorized for the target resource. This results in an access control flaw. Testers can easily manipulate parameters to detect such flaws. Code analysis quickly shows whether authorization is correct.		Such flaws can compromise all the functionality or data that is accessible. Unless references are unpredictable, or access control is enforced, data and functionality can be stolen, or abused.	Consider the business value of the exposed data and functionality. Also consider the business impact of public exposure of the vulnerability.



OWASP Top 10 – A5 Security Misconfiguration

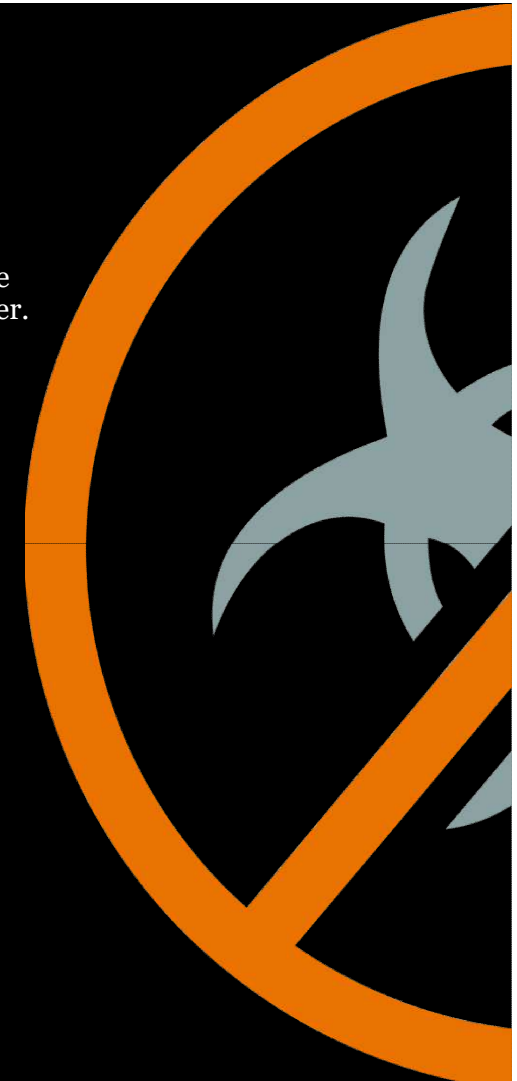
- Management, configuration and life-cycle of devices
- Generic examples:
 - Default accounts aren't changed
 - Directory listing is not disabled on server
- Not-directly related to web app, but widespread in embedded
 - TFTP/Telnet or other, inherently insecure protocol enabled without reason
 - OpenSSL weak ciphers -> «compatibility»
 - VPN but running on nullcipher or fail with setting default gateway

Threat Agents	Attack Vectors	Security Weakness	Technical Impacts	Business Impacts	
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anonymous external attackers as well as authorized users that may attempt to compromise the system. Also consider insiders wanting to disguise their actions.	Attackers access default accounts, unused pages, unpatched flaws, unprotected files and directories, etc. to gain unauthorized access to or knowledge of the system.	Security misconfiguration can happen at any level of an application stack, including the platform, web server, application server, database, frameworks, and custom code. Developers and system administrators need to work together to ensure that the entire stack is configured properly. Automated scanners are useful for detecting missing patches, misconfigurations, use of default accounts, unnecessary services, etc.	Such flaws frequently give attackers unauthorized access to some system data or functionality. Occasionally, such flaws result in a complete system compromise.	The system could be completely compromised without you knowing it. All of your data could be stolen or modified slowly over time. Recovery costs could be expensive.	

OWASP Top 10 – A6 Sensitive data exposure

- EWON routers and gateways, ICS-CERT ICSA-15-351-03
- Passwords are passed in plain text allowing a malicious party to retrieve them from network traffic. The autocomplete setting of some eWON forms also allows these passwords to be retrieved from the browser. Compromise of the credentials would allow unauthenticated access.

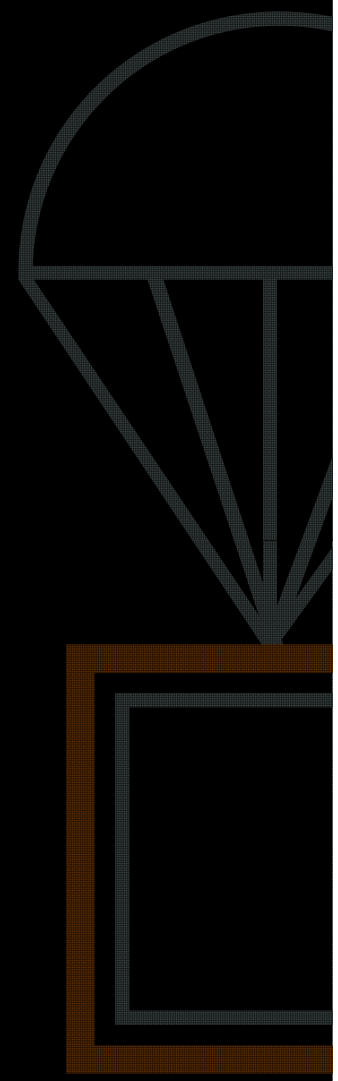
Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability DIFFICULT	Prevalence UNCOMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider who can gain access to your sensitive data and any backups of that data. This includes the data at rest, in transit, and even in your customers' browsers. Include both external and internal threats.	Attackers typically don't break crypto directly. They break something else, such as steal keys, do man-in-the-middle attacks, or steal clear text data off the server, while in transit, or from the user's browser.	The most common flaw is simply not encrypting sensitive data. When crypto is employed, weak key generation and management, and weak algorithm usage is common, particularly weak password hashing techniques. Browser weaknesses are very common and easy to detect, but hard to exploit on a large scale. External attackers have difficulty detecting server side flaws due to limited access and they are also usually hard to exploit.		Failure frequently compromises all data that should have been protected. Typically, this information includes sensitive data such as health records, credentials, personal data, credit cards, etc.	Consider the business value of the lost data and impact to your reputation. What is your legal liability if this data is exposed? Also consider the damage to your reputation.



OWASP Top 10 – A7 Insufficient Attack Protection

- Monitoring and management challenges, lack of NMS, IDS/IPS






Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact MODERATE	Application / Business Specific
Consider anyone with network access can send your application a request. Does your application detect and respond to both manual and automated attacks?	Attackers, known users or anonymous, send in attacks. Does the application or API detect the attack? How does it respond? Can it thwart attacks against known vulnerabilities?	Applications and APIs are attacked all the time. Most applications and APIs detect invalid input, but simply reject it, letting the attacker attack again and again. Such attacks indicate a malicious or compromised user probing or exploiting vulnerabilities. Detecting and blocking both manual and automated attacks, is one of the most effective ways to increase security. How quickly can you patch a critical vulnerability you just discovered?		Most successful attacks start with vulnerability probing. Allowing such probes to continue can raise the likelihood of successful exploit to 100%. Not quickly deploying patches aids attackers.	Consider the impact of insufficient attack protection on the business. Successful attacks may not be prevented, go undiscovered for long periods of time, and expand far beyond their initial footprint.



OWASP Top 10 – A8 Cross-Site Request Forgery

• Belden Gecko: ICSA 17-026-02A, Belden Security Bulletin 2017/7

• The Web interface of the Gecko does not verify that requests originate from the user.

 Threat Agents	 Attack Vectors	 Security Weakness		 Technical Impacts	 Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence UNCOMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who can load content into your users' browsers, and thus force them to submit a request to your website, including any website or other HTML feed that your users visit.	Attackers create forged HTTP requests and trick a victim into submitting them via image tags, iframes, XSS, or various other techniques. <u>If the user is authenticated</u> , the attack succeeds.	<u>CSRF</u> takes advantage of the fact that most web apps allow attackers to predict all the details of a particular action. Because browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones. Detection of CSRF flaws is fairly easy via penetration testing or code analysis.		Attackers can trick victims into performing any state changing operation the victim is authorized to perform (e.g., updating account details, making purchases, modifying data).	Consider the business value of the affected data or application functions. Imagine not being sure if users intended to take these actions. Consider the impact to your reputation.



OWASP Top 10 – A9 Using Components with Known Vulnerabilities






- Siemens, ICSA-14-105-03B, Siemens Industrial Products OpenSSL Heartbleed Vulnerability
- Other examples might be outdated PHP, Flash etc.

- Long life, software support and patching challenges

Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability AVERAGE	Impact MODERATE	Application / Business Specific
Some vulnerable components (e.g., framework libraries) can be identified and exploited with automated tools, expanding the threat agent pool beyond targeted attackers to include chaotic actors.	Attackers identify a weak component through scanning or manual analysis. They customize the exploit as needed and execute the attack. It gets more difficult if the used component is deep in the application.	Many applications and APIs have these issues because their development teams don't focus on ensuring their components and libraries are up to date. In some cases, the developers don't even know all the components they are using, never mind their versions. Component dependencies make things even worse. Tools are becoming commonly available to help detect components with known vulnerabilities.		The full range of weaknesses is possible, including injection, broken access control, XSS, etc. The impact could range from minimal to complete host takeover and data compromise.	Consider what each vulnerability might mean for the business controlled by the affected application. It could be trivial or it could mean complete compromise.

OWASP Top 10 – A10 Underprotected APIs (new)

- A top vulnerability we will live together for a long time
 - Mostly of this historical, compatibility and installed-base will force the support of compatibility-modes on protocols or support for outdated versions.
-
- Fallback to nullcipher if no common cipher suite has been found
 - Unencrypted protocols
 - M2M interfaces with no modifications expected.

 Threat Agents	 Attack Vectors	 Security Weakness		 Technical Impacts	 Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability DIFFICULT	Impact MODERATE	Application / Business Specific
Consider anyone with the ability to send requests to your APIs. Client software is easily reversed and communications are easily intercepted, so obscurity is no defense for APIs.	Attackers can reverse engineer APIs by examining client code, or simply monitoring communications. Some API vulnerabilities can be automatically discovered, others only by experts.	Modern web applications and APIs are increasingly composed of rich clients (browser, mobile, desktop) that connect to backend APIs (XML, JSON, RPC, GWT, custom). APIs (microservices, services, endpoints) can be vulnerable to the full range of attacks. Unfortunately, dynamic and sometimes even static tools don't work well on APIs, and they can be difficult to analyze manually, so these vulnerabilities are often undiscovered.		The full range of negative outcomes is possible, including data theft, corruption, and destruction; unauthorized access to the entire application; and complete host takeover.	Consider the impact of an API attack on the business. Does the API access critical data or functions? Many APIs are mission critical, so also consider the impact of denial of service attacks.

| Conclusion

No reason for special treatment for customer-facing services

- Use for example OWASP Top 10 tests to ensure common baseline with IT and OT
- There should be no device or service failing on Top 10





mnemonic
- *securing your business*

mnemonic