OWASP TOP TEN

Informaciona bezbednost

FAKULTET TEHNIČKIH NAUKA

V6

OWASP

OWASP - The Open Web Application Security Project

Neprofitna organizacija čija je misija da pruži pomoć i obuku programerima i kompanijama u cilju unapređenja bezbednosti

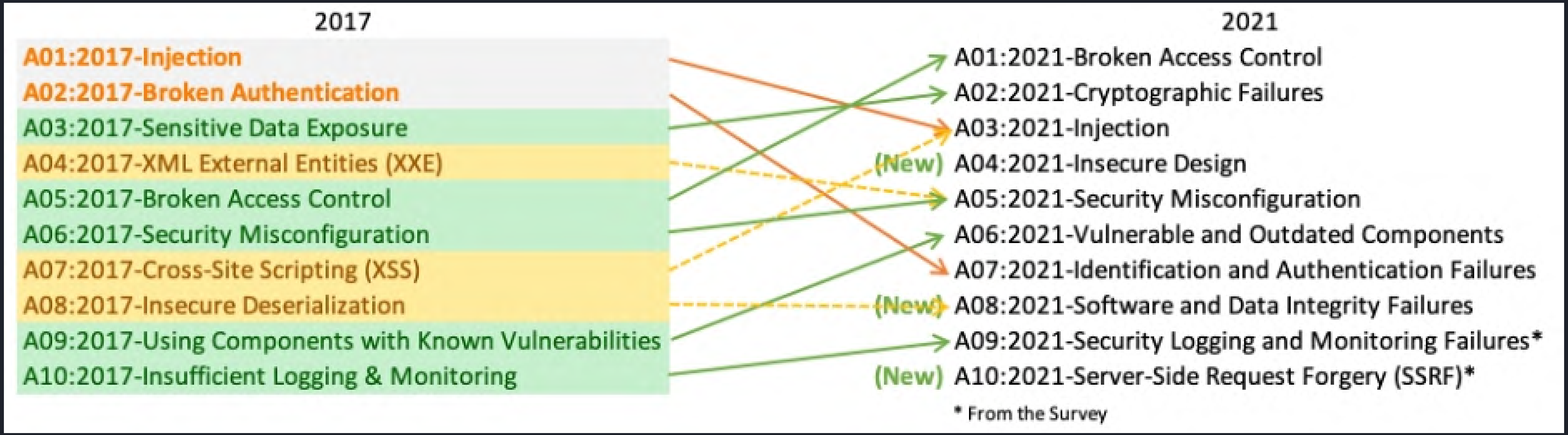
OWASP TOP 10

OWASP Top Ten je dokument koji sadrži listu najrasprostranjenijih i najozbiljnijih bezbednosnih problema u današnjim veb-baziranim sistemima

OWASP TOP 10

OWASP Top Ten je dokument koji sadrži listu najrasprostranjenijih i najozbiljnijih bezbednosnih problema u današnjim veb-baziranim sistemima

OWASP TOP 10



2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

OWASP TOP 10

2017

OWASP TOP 10

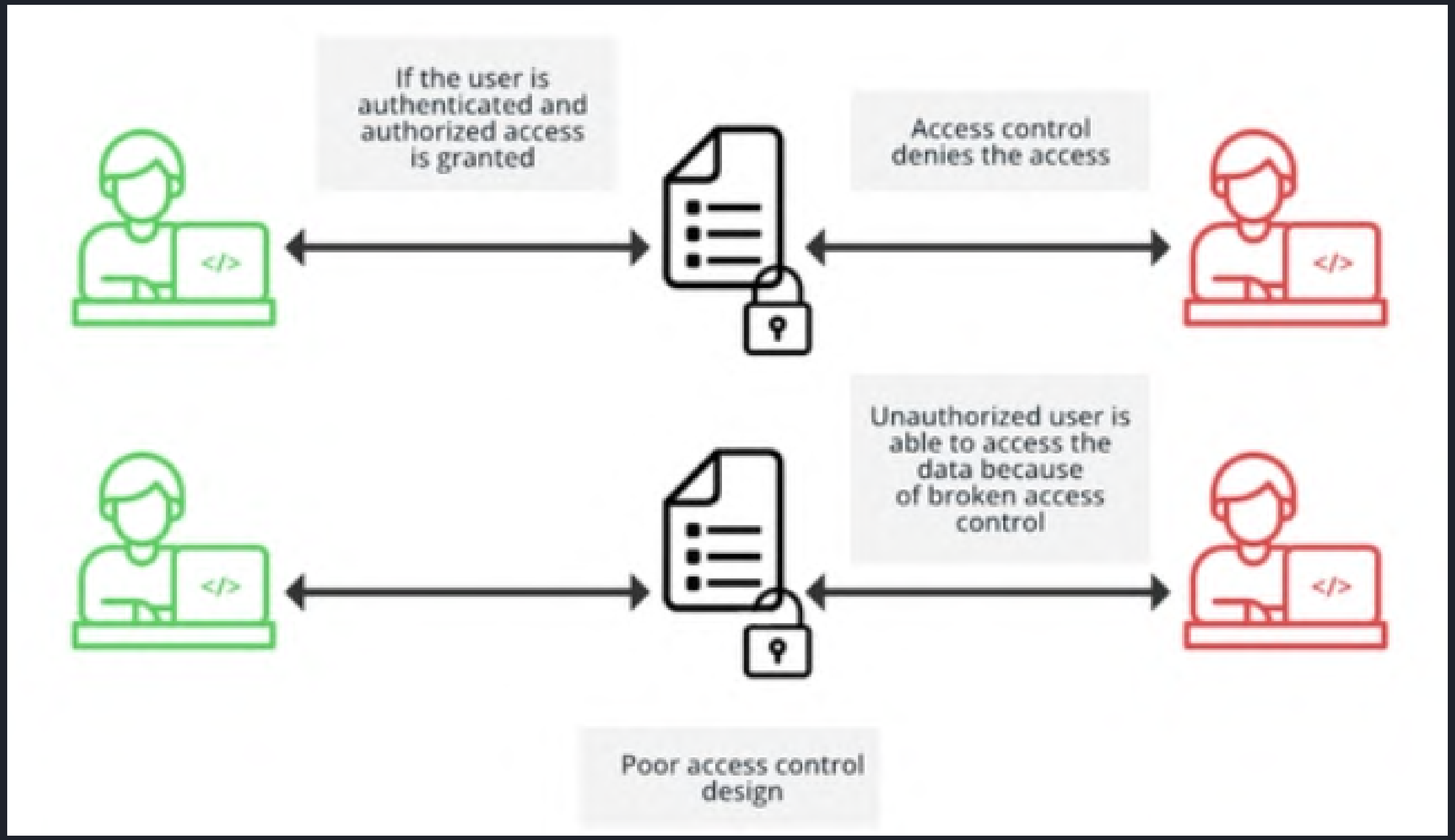
2021

OWASP TOP 10

01 BROKEN ACCESS CONTROL
02 CRYPTOGRAPHIC FAILURES
03 INJECTION
04 INSECURE DESIGN
05 SECURITY MISCONFIGURATION
06 VULNERABLE AND OUTDATED COMPONENTS
07 IDENTIFICATION AND AUTHENTICATION FAILURES
08 SOFTWARE AND DATA INTEGRITY FAILURES
09 SECURITY LOGGING AND MONITORING FAILURES
10 SERVER SIDE REQUEST FORGERY (SSRF)

01 BROKEN ACCESS CONTROL

- Korisnik je dobio pristup nekoj funkciji, a nije smeo
- Napadač je pristupio stranici koja je namenjena za administratora: https://example.com/app/admin_getappInfo



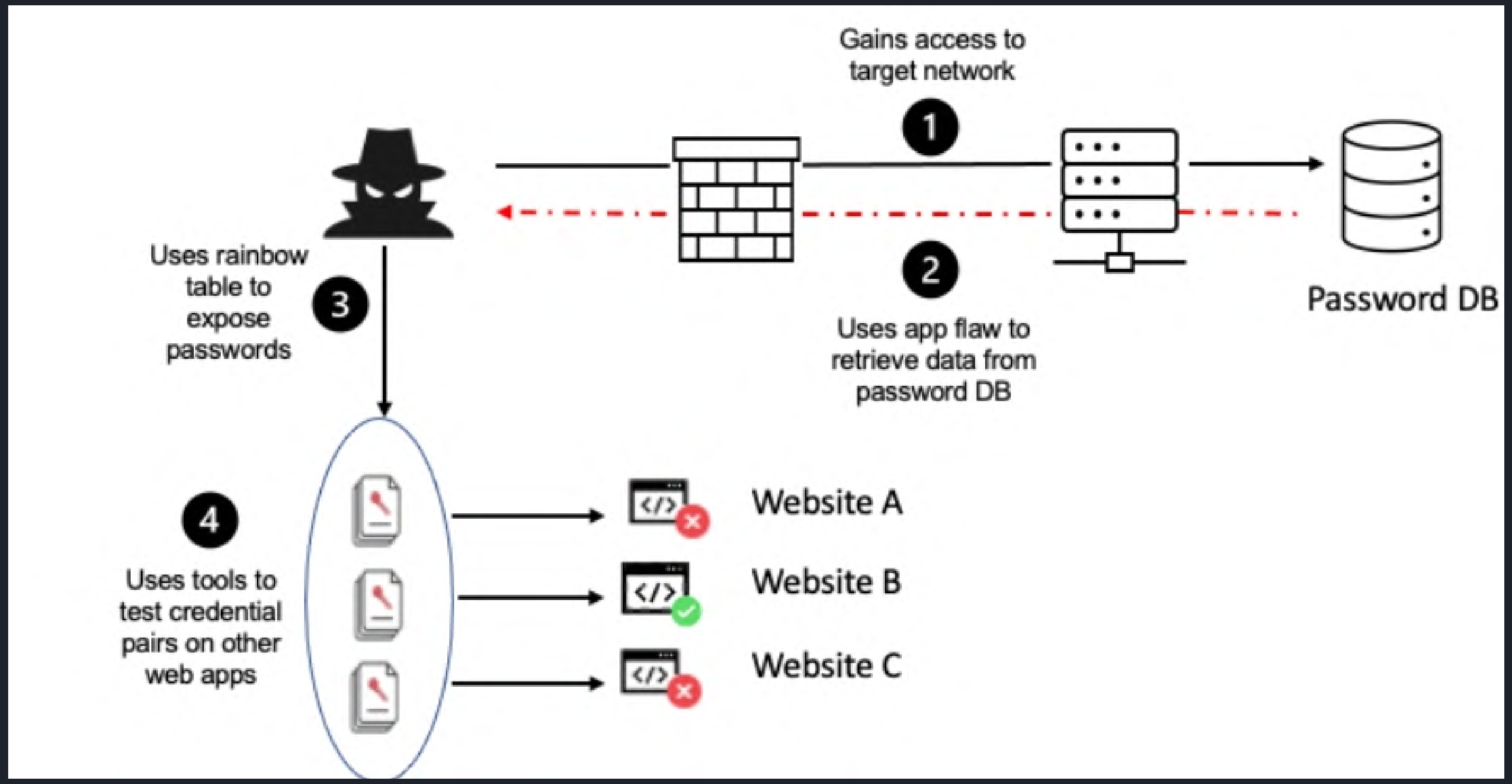
If the user is authenticated and authorized access is granted
Access control denies the access
</>
Unauthorized user is able to access the data because of broken access control
Poor access control design

HOW TO PREVENT?

- Deny access by default, except for public resources
- Build strong access control mechanisms and reuse them across the application
- Disable server directory listing and do not store sensitive data in root
- Rate limit API and controller access
- Validate JWT tokens after logout

02 CRYPTOGRAPHIC FAILURES

- Problem nastaje ako nije implementirana zaštita podataka u tranzitu i skladištu

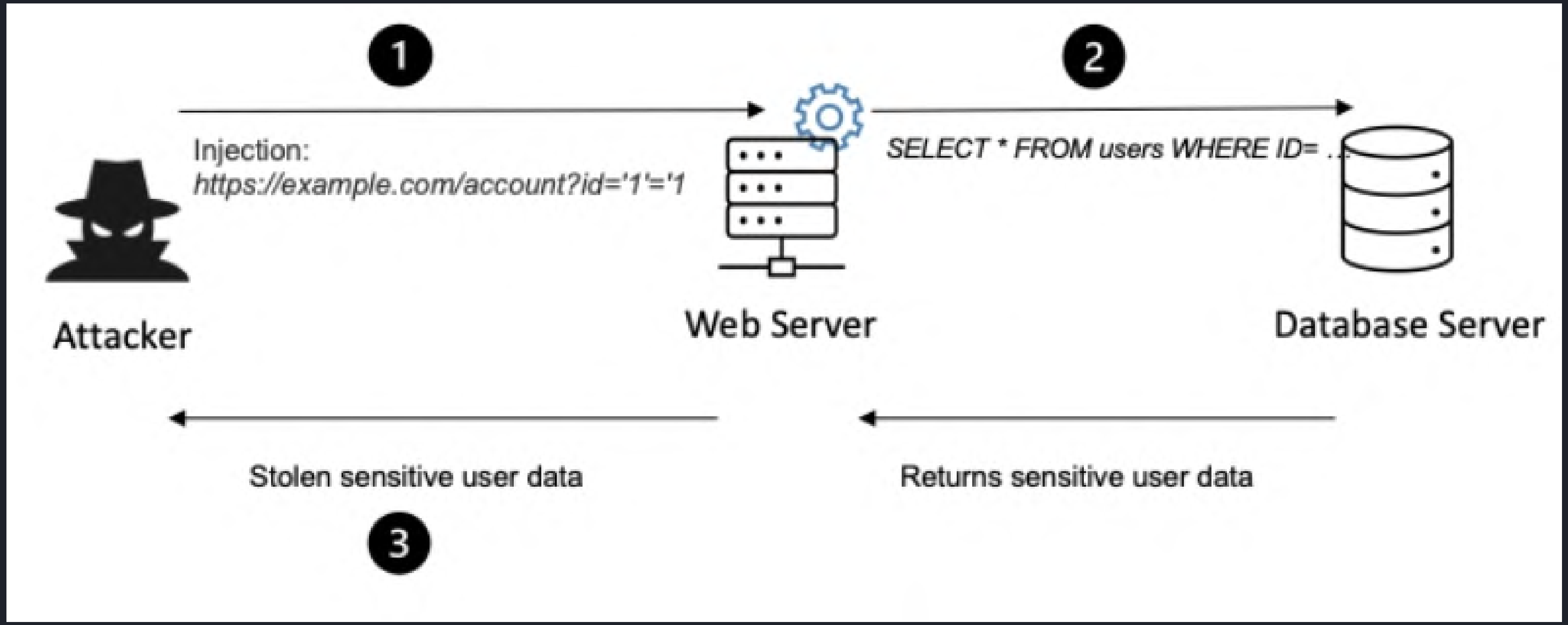

Gains access to target network
1
2
Uses app flaw to retrieve data from password DB
Password DB
3
Uses rainbow table to expose passwords
4
Uses tools to test credential pairs on other web apps
Website A
Website B
Website C

HOW TO PREVENT?

- Identify sensitive data and apply appropriate security controls
- Don’t store sensitive data unless absolutely needed — discard sensitive data, use tokenization or truncation
- Encrypt all sensitive data at rest using strong encryption algorithms, protocols and keys
- Encrypt data in transit using secure protocols like TLS and HTTP HSTS
- Disable caching for sensitive data
- Store passwords using strong, salted hashing functions like Argon2, scrypt and bcrypt

03 INJECTION

- Napadači ubrizgavaju kod u podatke koji se šalju interpreteru

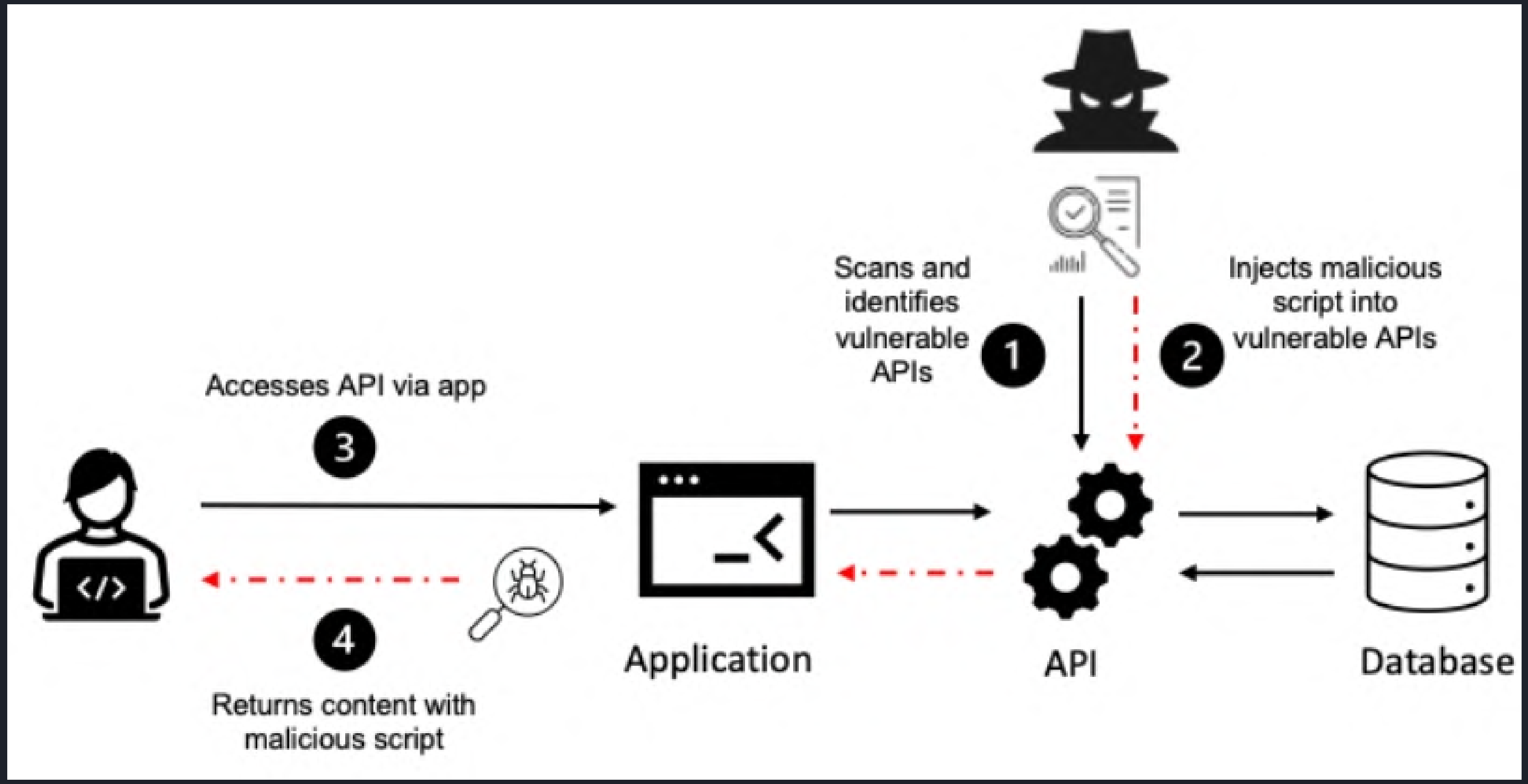

1
2
Injection:
https://example.com/account?id='1'='1
SELECT * FROM users WHERE ID= ...
Attacker
Web Server
Database Server
Stolen sensitive user data
Returns sensitive user data
3

HOW TO PREVENT?

- Use a safe API which avoids the use of the interpreter entirely
- Use positive or "whitelist" server-side input validation
- Escape special characters
- Use LIMIT and other SQL controls within queries to prevent mass disclosure of records in case of SQL injection.

04 INSECURE DESIGN

- Ranjivost se dešava usled nedostatka ili neefikasnosti određenih bezbednosnih kontrola


Scans and identifies vulnerable APIs
1
Injects malicious script into vulnerable APIs
2
Accesses API via app
3
4
Returns content with malicious script
Application
API
Database

HOW TO PREVENT?

- Establish a secure software development lifecycle (SSDLC)
- Leverage application security practices from early stages of software development
- Create a library of secure design patterns, and use it to build new applications
- Leverage threat modeling to design critical features like authentication and access control

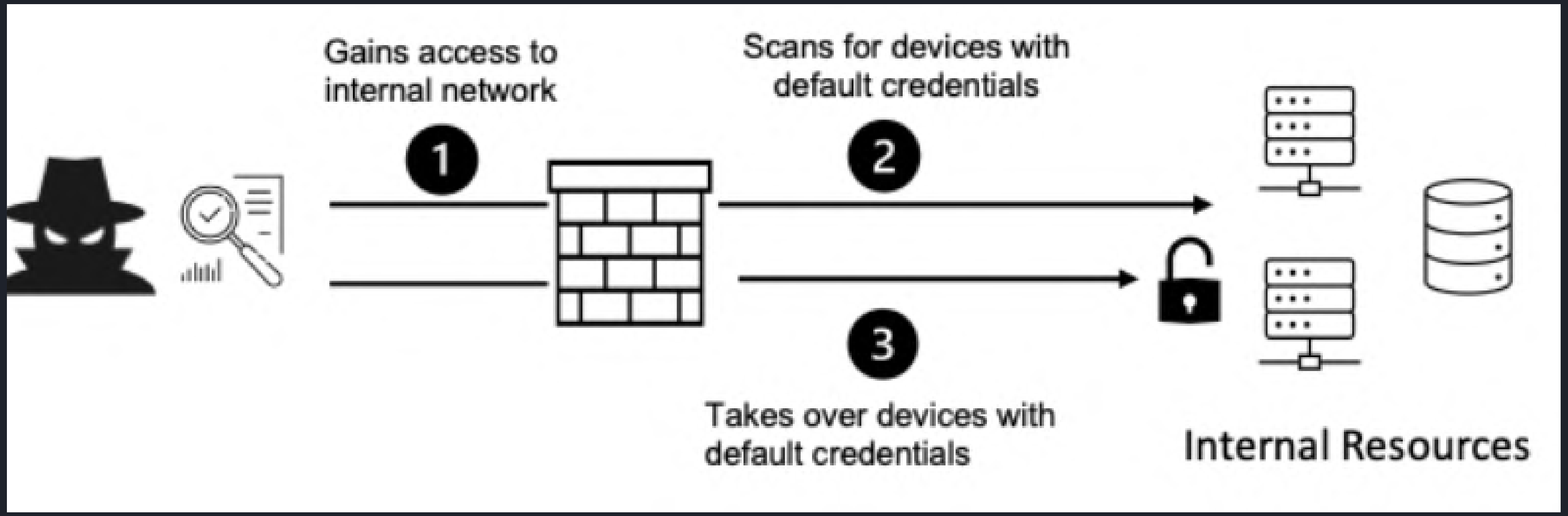
Integrate security concerns and controls into all user stories

If you think good architecture is expensive, try bad architecture.

Brian Foote & Joseph Yo

05 SECURITY MISCONFIGURATION

- Loša konfiguracija

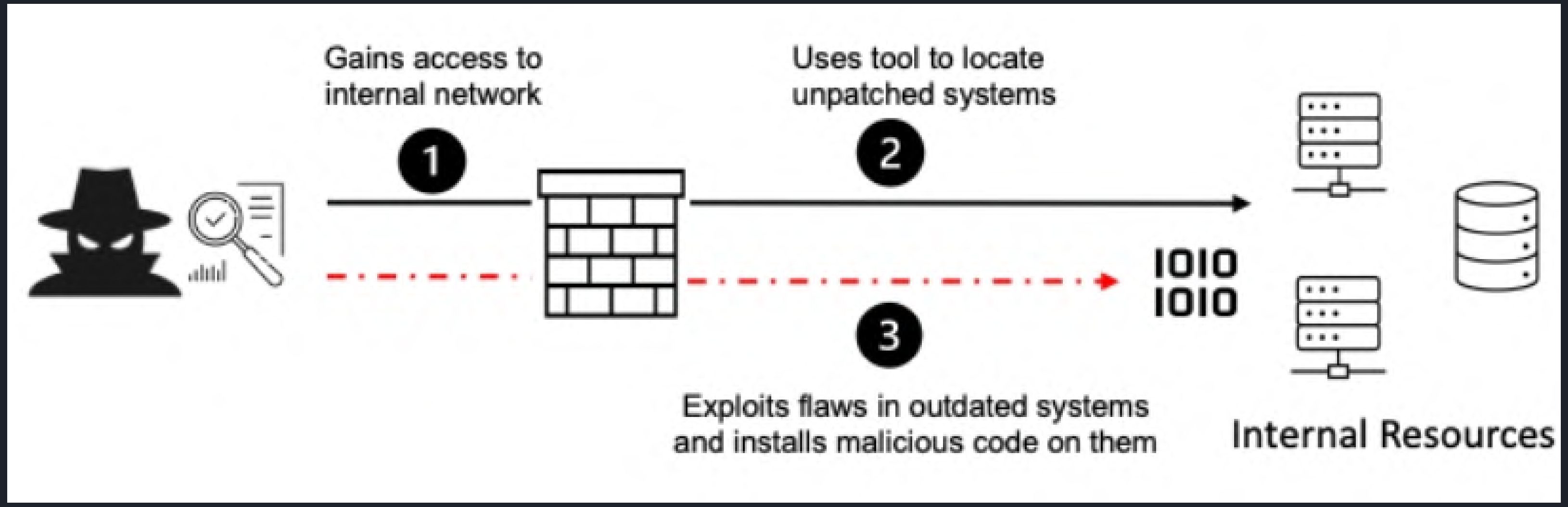

Gains access to internal network
1
Scans for devices with default credentials
2
3
Takes over devices with default credentials
Internal Resources

HOW TO PREVENT?

- Establish a hardening process for applications, which is fast and easy to deploy
- Configure development, QA, and production identically (with different credentials)
- All systems should have a minimal setup without unnecessary features and components
- Configurations should be regularly updated, applying patches and security advisories
- Establish an automated process to verify secure configurations in all environments

06 VULNERABLE AND OUTDATED COMPONENTS

- Upotreba komponenti, biblioteka koje imaju ranjivosti i zastarele su

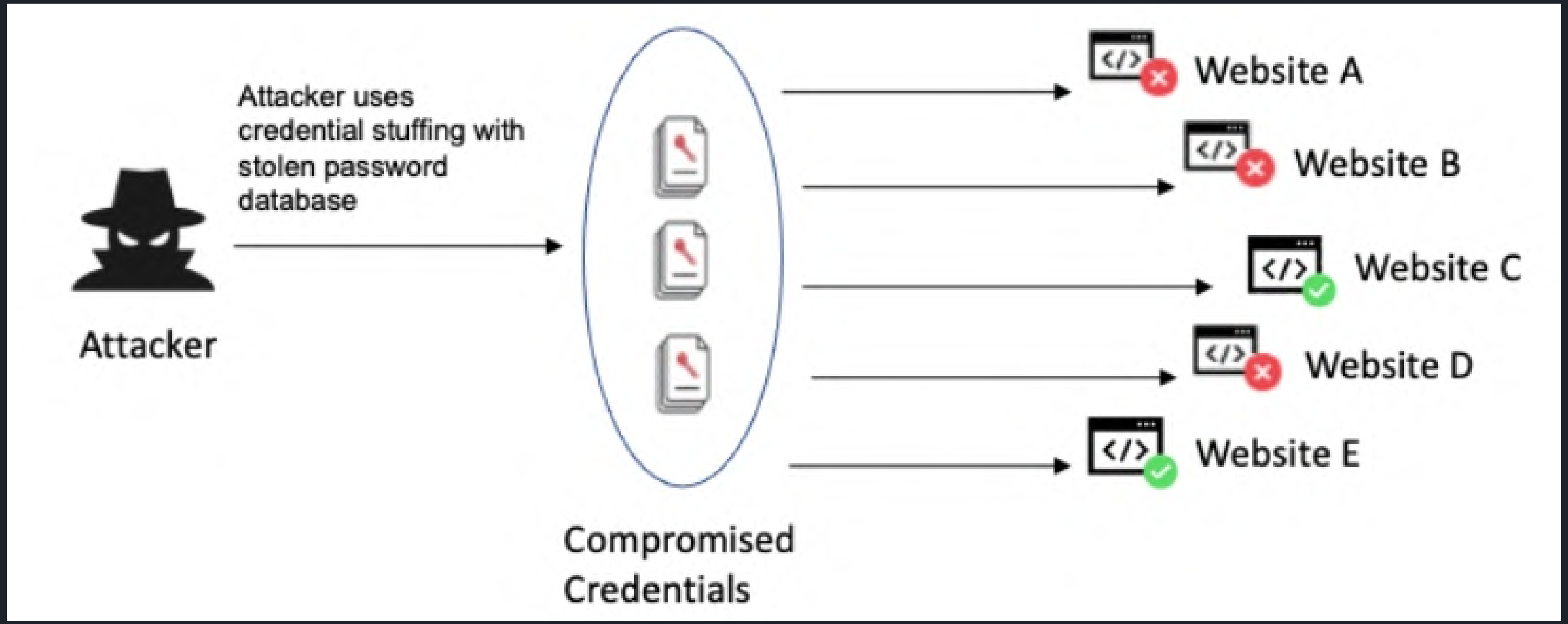

Gains access to internal network
1
Uses tool to locate unpatched systems
2
1010
1010
3
Exploits flaws in outdated systems and installs malicious code on them
Internal Resources

HOW TO PREVENT?

- Remove unused dependencies, features, components, and files from applications.
- Maintain an inventory of components and their versions, both on the client side and server side, using software composition analysis (SCA) tools
- Continuously scan libraries and their dependencies for vulnerable components
- Only use components from official sources, and prefer signed packages
- Urgently remediate vulnerabilities, remove affected components, or apply a virtual patch

07 IDENTIFICATION AND AUTHENTICATION FAILURES

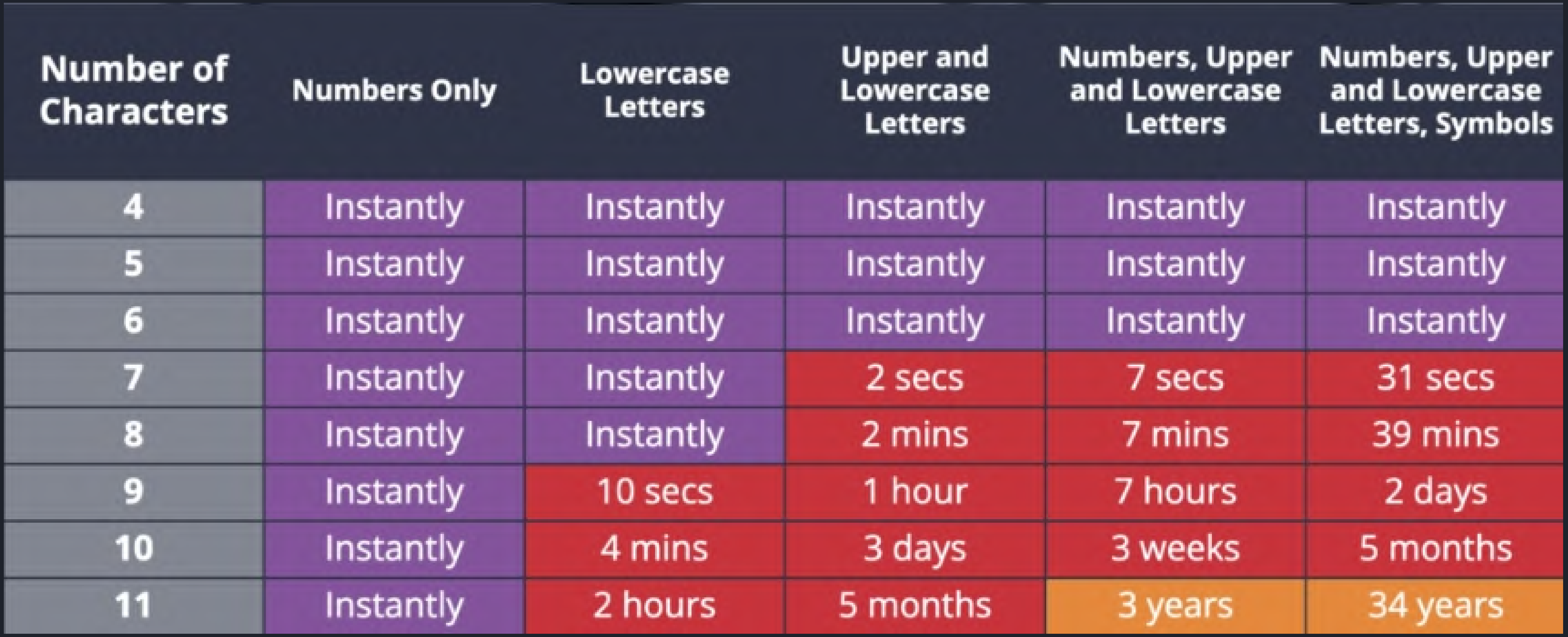
- Autentifikacija nije dobro implementirana



HOW TO PREVENT?

- Implement multi-factor authentication
- Do not deploy systems with default credentials
- Check for a list of the top 10,000 worst passwords
- Use the guidelines in NIST 800-63 B section 5.1.1 for Memorized Secrets
- Harden all authentication-related processes like registration and credential recovery
- Limit or delay failed login attempts

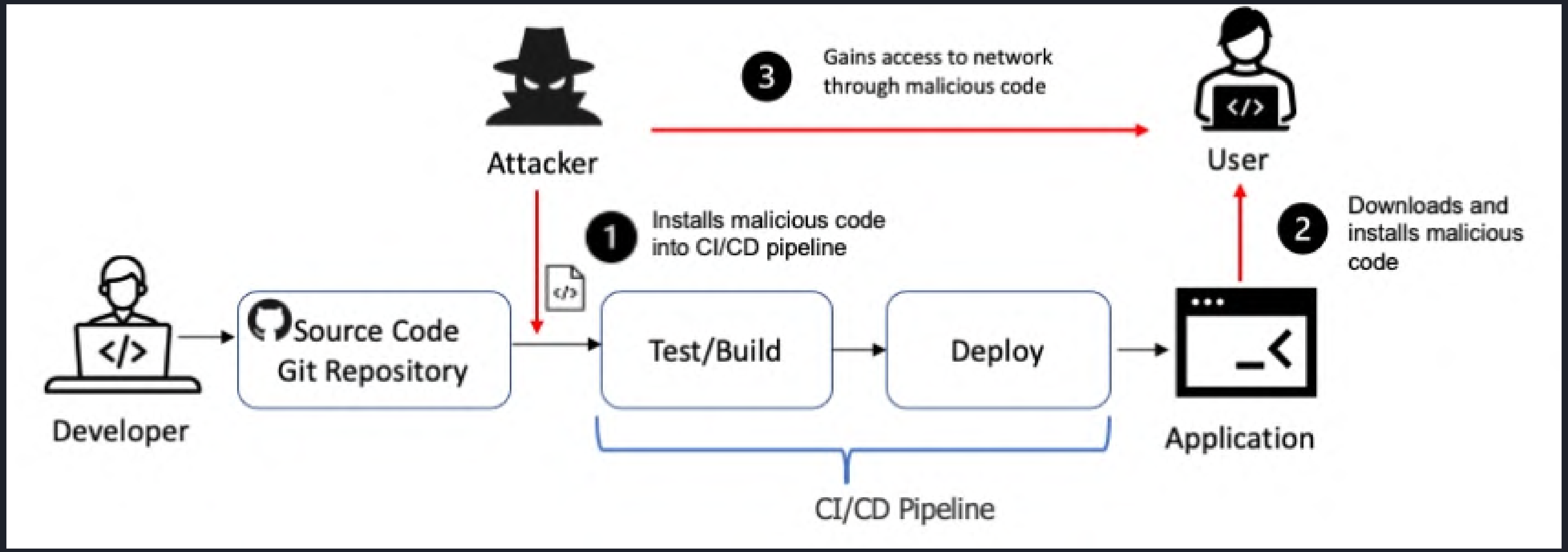
07 IDENTIFICATION AND AUTHENTICATION FAILURES



Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	2 secs	7 secs	31 secs
8	Instantly	Instantly	2 mins	7 mins	39 mins
9	Instantly	10 secs	1 hour	7 hours	2 days
10	Instantly	4 mins	3 days	3 weeks	5 months
11	Instantly	2 hours	5 months	3 years	34 years

08 SOFTWARE AND DATA INTEGRITY FAILURES

- Kod i infrastruktura koji su podložni kršenju integriteta

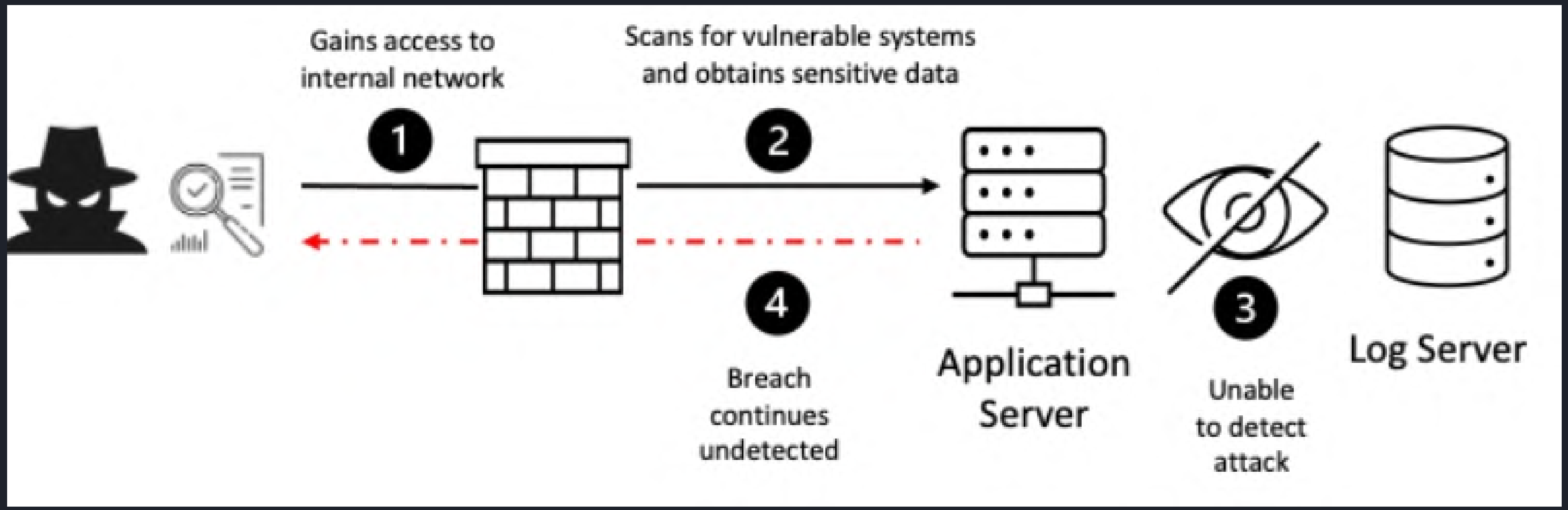

Attacker
3 Gains access to network through malicious code
User
1 Installs malicious code into CI/CD pipeline
2 Downloads and installs malicious code
Developer
Source Code Git Repository
Test/Build
Deploy
Application
CI/CD Pipeline

HOW TO PREVENT?

- Use digital signatures or similar mechanisms to verify software or data is from the expected source and has not been altered
- Ensure libraries and dependencies, such as npm or maven, are pulling from trusted repositories
- Establish a review process for code and configuration changes
- Ensure that your CI/CD pipeline has proper configuration and access controls

09 SECURITY LOGGING AND MONITORING FAILURES

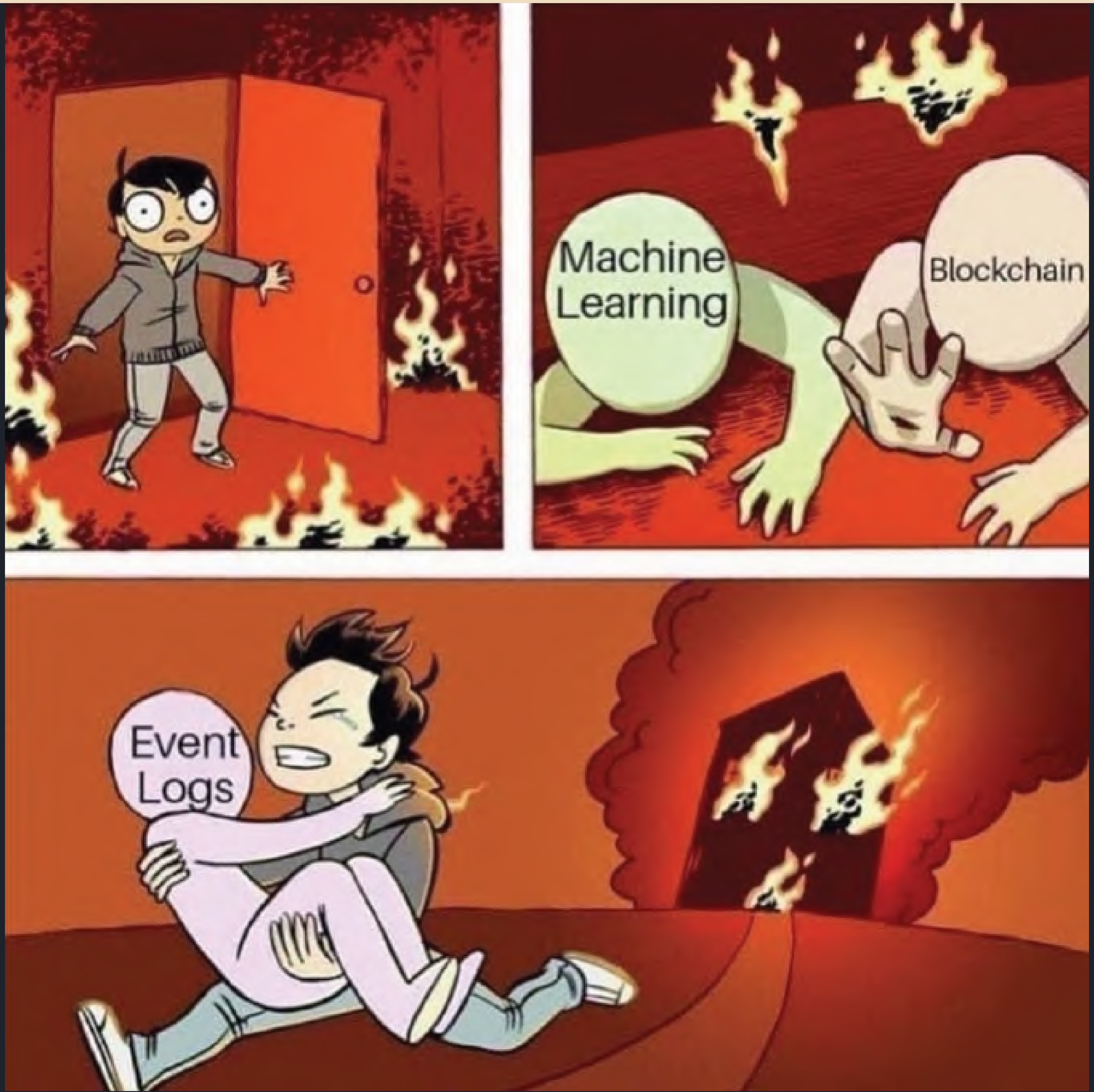
- Implementirati loging i monitoring


Gains access to internal network
1
Scans for vulnerable systems and obtains sensitive data
2
4
Breach continues undetected
Application Server
3
Unable to detect attack
Log Server

HOW TO PREVENT?

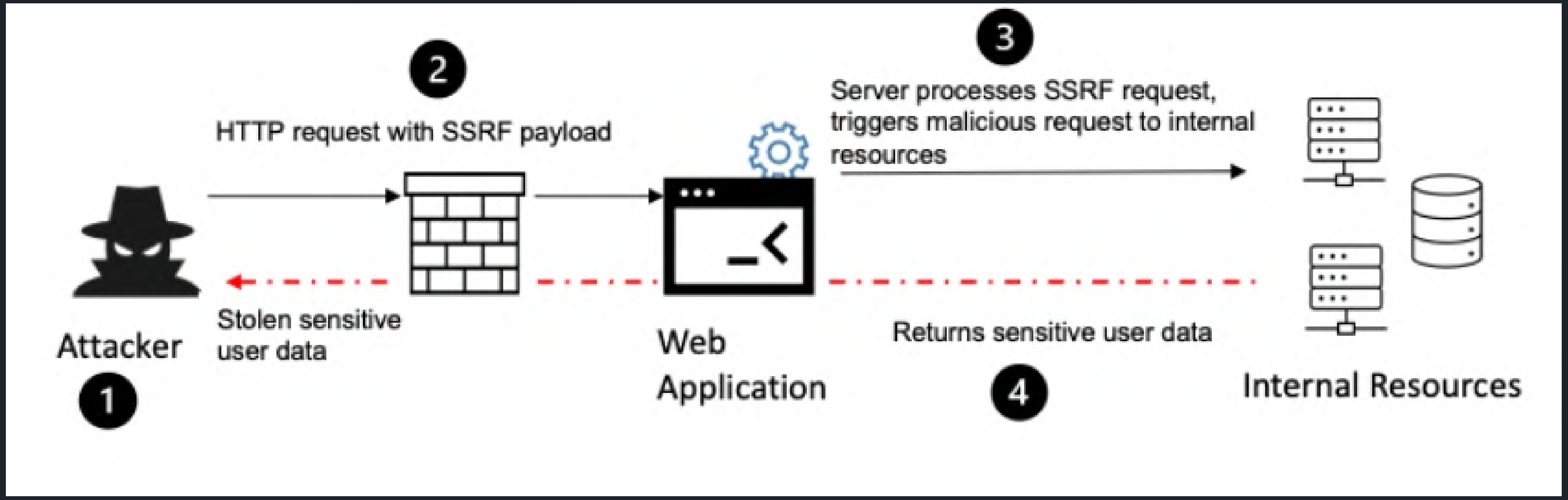
- Ensure login, access control, and server-side input validation is logged
- Ensure logs contain enough context to identify suspicious behavior and enable in-depth forensic analysis
- Ensure logs are in a format compatible with log management solutions
- Take measures to prevent attackers from tampering with log data

09 SECURITY LOGGING AND MONITORING FAILURES



10 SERVER SIDE REQUEST FORGERY (SSRF)

- Problem nastaje kada aplikacija povuče podatke bez prethodne validacije URL-a


2
HTTP request with SSRF payload
3
Server processes SSRF request, triggers malicious request to internal resources
Attacker
1
Stolen sensitive user data
Web Application
Returns sensitive user data
4
Internal Resources

HOW TO PREVENT?

- Avoid accepting URLs in client inputs, and if absolutely necessary, sanitize inputs
- Isolate any remote resource access functionality in a separate network to reduce impact
- Use "deny by default" firewall policies to block unwanted Internet traffic
- Use a positive allow list with URL schema, port, and destination
- Disable HTTP redirections
- Never return raw responses to clients