Vulnerabilities|Att@ks|Threats|Checklists In-details [OWASP, SANS, NIST, MITRE|CWE]

Server Security Misconfiguration

- ✓ Server-Side Request Forgery (SSRF)
  - Internal High Impact
  - Internal Scan and/or Medium Impact
  - External- Low impact
  - External-DNS Query Only
- ✓ Unsafe Cross-Origin Resource Sharing
- ✓ HTTP Request Smuggling
- ✓ Path Traversal
- ✓ Directory Listing Enabled
  - Sensitive Data Exposure
  - Non-Sensitive Data Exposure
- ✓ Same-Site Scripting
- ✓ SSL Attack (BREACH, POODLE etc.)
- ✓ Using Default Credentials
- ✓ Misconfigured DNS
  - Basic Subdomain Takeover
  - High Impact Subdomain Takeover
  - Zone Transfer
  - Missing Certification Authority Authorization (CAA) Record
- ✓ Mail Server Misconfiguration
  - No Spoofing Protection on Email Domain
  - Email Spoofing to Inbox due to Missing or Misconfigured DMARC on Email Domain
  - Email Spoofing to Spam Folder
  - Missing or Misconfigured SPF and/or DKIM
  - Email Spoofing on Non-Email Domain
- ✓ Database Management System (DBMS) Misconfiguration
  - Excessively Privileged User/DBA
- ✓ Lack of Password Confirmation
  - Change Email Address
  - Change Password
  - Delete Account
  - Manage 2FA
- ✓ No Rate Limiting on Form
  - Registration
  - Login
  - Email-Triggering
  - SMS-Triggering
  - Change Password
- ✓ Unsafe File Upload
  - No Antivirus
  - No Size Limit
  - File Extension Filter Bypass
- ✓ Cookie Scoped to Parent Domain
- ✓ Missing Secure or HTTPOnly Cookie Flag
  - Session Token
  - Non-Session Cookie
- ✓ Clickjacking
  - Sensitive Click-Based Action
  - Form Input
  - Non-Sensitive Action
- ✓ OAuth Misconfiguration
  - Account Takeover
  - Account Squatting
  - Missing/Broken State Parameter
  - Insecure Redirect URI
- ✓ CAPTCHA
  - Implementation Vulnerability
  - Brute Force
  - Missing
- ✓ Exposed Admin Portal
  - To Internet
- ✓ Missing DNSSEC
- ✓ Fingerprinting/Banner Disclosure
- ✓ Username/Email Enumeration
  - Brute Force
- ✓ Potentially Unsafe HTTP Method Enabled
  - OPTIONS
  - TRACE
- ✓ Insecure SSL
  - Lack of Forward Secrecy
  - Insecure Cipher Suite
  - Certificate Error
- ✓ Reflected File Download (RFD)
- ✓ Lack of Security Headers
  - X-Frame-Options
  - Cache-Control for a Non-Sensitive Page
  - X-XSS-Protection
  - Strict-Transport-Security
  - X-Content-Type-Options
  - Content-Security-Policy
  - Public-Key-Pins
  - X-Content-Security-Policy
  - X-Webkit-CSP
  - Content-Security-Policy-Report-Only
  - Cache-Control for a Sensitive Page
- ✓ Web Application Firewall (WAF) Bypass
  - Direct Server Access
- ✓ Race Condition
- ✓ Cache Poisoning
- ✓ Bitsquatting

Server-Side Injection

- ✓ File Inclusion
  - Local
- ✓ Parameter Pollution
  - Social Media Sharing Buttons
- ✓ Remote Code Execution (RCE)
- ✓ LDAP Injection
- ✓ SQL Injection
- ✓ XML External Entity Injection (XXE)
- ✓ HTTP Response Manipulation
  - Response Splitting (CRLF)
- ✓ Content Spoofing
  - iframe Injection
  - Impersonation via Broken Link Hijacking
  - External Authentication Injection
  - Flash-Based External Authentication Injection
  - HTML Content Injection
  - Email HTML Injection
  - Email Hyperlink Injection Based on Email Provider
  - Text Injection
  - Homograph/IDN-Based
  - Right-to-Left Override (RTLO)
- ✓ Server-Side Template Injection (SSTI)
  - Basic
  - Custom

Broken Authentication and Session Management

- ✓ Authentication Bypass
- ✓ Second Factor Authentication (2FA) Bypass
- ✓ Privilege Escalation
- ✓ Cleartext Transmission of Session Token
- ✓ Weak Login Function
  - Not Operation or Intended Public Access
  - Other Plaintext Protocol with no Secure Alternative
  - Over HTTP
- ✓ Session Fixation
  - Remote Attack Vector
  - Local Attack Vector

- ✓ Failure to Invalidate Session
  - On Logout (Client and Server-Side)
  - On Permission Change
  - On Logout (Server-Side Only)
  - On Password Reset and/or Change
  - Concurrent Sessions On Logout
  - On Email Change
  - On 2FA Activation/Change
  - Long Timeout
- ✓ Concurrent Logins
- ✓ Weak Registration Implementation
  - Over HTTP

Sensitive Data Exposure

- ✓ Disclosure of Secrets
  - For Publicly Accessible Asset
  - Personally Identifiable Information (PII) Leakage/Exposure
  - For Internal Asset
  - Pay-per-Use Abuse
  - Intentionally Public, Sample, or Invalid
  - Data/Traffic Spam
  - Non-Corporate User
- ✓ EXIF Geolocation Data Not Stripped from Uploaded Images
  - Automatic User Enumeration
  - Manual User Enumeration
- ✓ Visible Detailed Error/Debug Page
  - Detailed Server Configuration
  - Full Path Disclosure
  - Descriptive Stack Trace
- ✓ Disclosure of Known Public Information
- ✓ Token Leakage via Referrer
  - Trusted 3rd Party
  - Untrusted 3rd Party
  - Over HTTP
- ✓ Sensitive Token in URL
  - User Facing
  - In the Background
  - On Password Reset
- ✓ Non-Sensitive Token in URL
- ✓ Weak Password Reset Implementation
  - Password Reset Token Sent Over HTTP
  - Token Leakage via Host Header Poisoning
- ✓ Mixed Content (HTTPS Sourcing HTTP)
- ✓ Sensitive Data Hardcoded
  - OAuth Secret
  - File Paths
- ✓ Internal IP Disclosure
- ✓ Cross-Site Script Inclusion (XSSI)
- ✓ JSON Hijacking
- ✓ Via localStorage/SessionStorage
  - Sensitive Token
  - Non-Sensitive Token

Cross-Site Scripting (XSS)

- ✓ Stored
  - Non-Privileged User to Anyone
  - Privileged User to Privilege Elevation
  - Privileged User to No Privilege Elevation
  - CSRF/URL-Based
  - Self
- ✓ Reflected
  - Non-Self
  - Self
- ✓ Flash-Based
- ✓ Cookie-Based
- ✓ IE-Only
- ✓ Referrer
- ✓ TRACE Method
- ✓ Universal (UXSS)
- ✓ Off-Domain
  - Data URI

Broken Access Control(BAC)

- ✓ Insecure Direct Object References (IDOR)
  - Read/Edit/Delete Non-Sensitive Information
  - Read/Edit/Delete Sensitive Information/Complex Object Identifiers (GUID)
  - Read Sensitive Information/Iterable Object Identifiers
  - Edit/Delete Sensitive Information/Iterable Object Identifiers
  - Read/Edit/Delete Sensitive Information/Iterable Object Identifiers
- ✓ Username/Email Enumeration
  - Non-Brute Force
- ✓ Exposed Sensitive Android Intent
- ✓ Exposed Sensitive iOS URL Scheme

Cross-Site Request Forgery (CSRF)

- ✓ Application-Wide
- ✓ Action-Specific
  - Authenticated Action
  - Unauthenticated Action
  - Logout
- ✓ CSRF Token Not Unique Per Request
- ✓ Flash-Based

Application-Level Denial-of-Service (DoS)

- ✓ Excessive Resource Consumption
  - Injection (Prompt)
- ✓ Critical Impact and/or Easy Difficulty
- ✓ High Impact and/or Medium Difficulty
- ✓ App Crash
  - Malformed Android Intents
  - Malformed iOS URL Schemes

Unvalidated Redirects and Forwards

- ✓ Open Redirect
  - GET-Based
  - POST-Based
  - Header-Based
  - Flash-Based
- ✓ Tabnabbing
- ✓ Lack of Security Speed Bump Page

External Behaviour

- ✓ Browser Feature
  - Plaintext Password Field
  - Save Password
  - Autocomplete Enabled
  - Autocorrect Enabled
  - Aggressive Offline Caching
- ✓ CSV Injection
- ✓ Captcha Bypass
  - Crowdsourcing
- ✓ System Clipboard Leak
  - Shared Links
- ✓ User Password Persisted in Memory

Insufficient Security Configurability

- ✓ Weak Password Policy
- ✓ No Password Policy
- ✓ Password Policy Bypass
- ✓ Weak Password Reset Implementation
  - Token is Not Invalidated After Use
  - Token is Not Invalidated After Email Change
  - Token is Not Invalidated After Password Change
  - Token Has Long Timed Expiry
  - Token is Not Invalidated After New Token is Requested
  - Token is Not Invalidated After Login
- ✓ Verification of Contact Method not Required
- ✓ Lack of Notification Email
- ✓ Weak Registration Implementation
  - Allows Disposable Email Addresses
- ✓ Weak 2FA Implementation
  - 2FA Secret Cannot be Rotated
  - 2FA Secret Remains Obtainable After 2FA is Enabled
  - Missing Failsafe
  - 2FA Code is Not Updated After New Code is Requested
  - Old 2FA Code is Not Invalidated After New Code is Generated

Using Components with Known Vulnerabilities

- ✓ Rosetta Flash
- ✓ Outdated Software Version
- ✓ Captcha Bypass
  - OCR (Optical Character Recognition)

Insecure Data Storage

- ✓ Sensitive Application Data Stored Unencrypted
  - On External Storage
  - On Internal Storage
- ✓ Server-Side Credentials Storage
  - Plaintext
- ✓ Non-Sensitive Application Data Stored Unencrypted
- ✓ Screen Caching Enabled

Lack of Binary Hardening

- ✓ Lack of Exploit Mitigations
- ✓ Lack of Jailbreak Detection
- ✓ Lack of Obfuscation
- ✓ Runtime Instrumentation-Based

Insecure Data Transport

- ✓ Cleartext Transmission of Sensitive Data
- ✓ Executable Download
  - No Secure Integrity Check
  - Secure Integrity Check

Insecure OS/Firmware

- ✓ Command Injection
- ✓ Hardcoded Password
  - Privileged User
  - Non-Privileged User

Cryptographic Weakness

- ✓ Insufficient Entropy
  - Limited Random Number Generator (RNG) Entropy Source
  - Use of True Random Number Generator (TRNG) for Non-Security Purpose
  - Pseudo-Random Number Generator (PRNG) Seed Reuse
  - Predictable Pseudo-Random Number Generator (PRNG) Seed
  - Small Seed Space in Pseudo-Random Number Generator (PRNG)
  - Initialization Vector (IV) Reuse
  - Predictable Initialization Vector (IV)
  - Insecure Implementation
  - Missing Cryptographic Step
  - Improper Following of Specification (Other)
- ✓ Weak Hash
  - Lack of Salt
  - Use of Predictable Salt
  - Predictable Hash Collision
- ✓ Insufficient Verification of Data Authenticity
  - Integrity Check Value (ICV)
  - Cryptographic Signature
- ✓ Insecure Key Generation
  - Improper Asymmetric Prime Selection
  - Improper Asymmetric Exponent Selection
  - Insufficient Key Stretching
  - Insufficient Key Space
  - Key Exchange Without Entity Authentication
- ✓ Key Reuse
  - Lack of Perfect Forward Secrecy
  - Intra-Environment
  - Inter-Environment
- ✓ Broken Cryptography
  - Use of Broken Cryptographic Primitive
  - Use of Vulnerable Cryptographic Library
- ✓ Side-Channel Attack
  - Padding Oracle Attack
  - Timing Attack
  - Power Analysis Attack
  - Emanations Attack
  - Differential Fault Analysis
- ✓ Use of Expired Cryptographic Key (or Certificate)
- ✓ Incomplete Cleanup of Keying Material

Privacy Concerns

- ✓ Unnecessary Data Collection
  - WiFi SSID+Password

Network Security Misconfiguration

- ✓ Telnet Enabled

Mobile Security Misconfiguration

- ✓ SSL Certificate Pinning
  - Absent
  - Defeatable
- ✓ Tapjacking
- ✓ Clipboard Enabled
- ✓ Auto Backup Allowed by Default

Client-Side Injection

- ✓ Binary Planting
  - Default Folder Privilege Escalation
  - Non-Default Folder Privilege Escalation
  - No Privilege Escalation

Automotive Security Misconfiguration

- ✓ Infotainment, Radio Head Unit
  - Sensitive data Leakage/Exposure
  - OTA Firmware Manipulation
  - Code Execution (CAN Bus Pivot)
  - Code Execution (No CAN Bus Pivot)
  - Unauthorized Access to Services (API/Endpoints)

- Source Code Dump
- Denial of Service (DoS/Brick)
- Default Credentials

✓ RF Hub
- Key Fob Cloning
- CAN Injection/Interaction
- Data Leakage/Pull Encryption Mechanism
- Unauthorized Access/Turn On
- Roll Jam
- Replay
- Relay

✓ CAN
- Injection (Battery Management System)
- Injection (Steering Control)
- Injection (Pyrotechnical Device Deployment Tool)
- Injection (Headlights)
- Injection (Sensors)
- Injection (Vehicle Anti-theft Systems)
- Injection (Powertrain)
- Injection (Basic Safety Message)
- Injection (Disallowed Messages)
- Injection (DoS)

✓ Battery Management System
- Firmware Dump
- Fraudulent Interface

✓ GNSS/GPS
- Spoofing

✓ Immobilizer
- Engine Start

✓ Automatic Braking System (ABS)
- Unintended Acceleration/Brake

✓ Roadside Unit (RSU)
- Sybil Attack

AI Application Security

✓ Large Language Model (LLM) Security
- Prompt Injection
- LLM Output Handling
- Training Data Poisoning
- Excessive Agency/Permission Manipulation

Security Testing Checklists

✓ **100 web exploits**

Injection attacks
- SQL Injection
- NoSQL Injection
- Command Injection
- LDAP Injection
- XML Injection
- XPath Injection
- Server-Side Template Injection (SSTI)
- Code Injection
- Log Injection
- CRLF Injection

Cross-Site Scripting (XSS)
- Stored XSS
- Reflected XSS
- DOM-based XSS

Cross-Site Request Forgery (CSRF)

Authentication and Authorization attacks
- Broken Authentication
- Broken Access Control
- Privilege Escalation
- Insecure Direct Object Reference (IDOR)
- Missing Authentication for Critical Function
- Password-related attacks
- Brute Force attacks
- Credential Stuffing
- Password Spraying
- Dictionary attacks

Session Management attacks
- Session Fixation
- Session Hijacking
- Session Timeout

Insecure Deserialization

Security Misconfiguration

Sensitive Data Exposure
- Insecure Storage of Sensitive Data
- Information Leakage and Improper Error Handling
- Insecure Data Transfer

Using Components with Known Vulnerabilities

Insufficient Logging and Monitoring

Unvalidated Redirects and Forwards

XML External Entity (XXE) attacks

Clickjacking (UI Redressing)

Server-Side Request Forgery (SSRF)

Insecure File Upload

Path Traversal (Directory Traversal)

Remote Code Execution (RCE)

Remote File Inclusion (RFI)

Local File Inclusion (LFI)

HTTP Request Smuggling

HTTP Response Splitting

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks
- Application-Level DoS
- Network-Level DoS

Insecure CORS (Cross-Origin Resource Sharing) Configuration

Insecure WebSocket Implementation

Cache Poisoning

Subdomain Takeover

Race Condition

Insecure APIs

Insecure Cryptographic Practices

Insecure Password Recovery Mechanisms

Insecure Serverless Functions

Insecure Docker Container Configuration

Insecure Kubernetes Configuration

Web Application Firewall (WAF) Bypass

Open Redirects

Man-in-the-Middle (MITM) attacks

Header Injection

Host Header Injection

Reverse Tabnabbing

Parameter Tampering

Timing Attacks

Memory Leaks

Same-Origin Method Execution (SOME)

Insecure Object References

Insecure WebSockets

Cryptojacking

Content Security Policy (CSP) Bypass

MIME Sniffing

Insecure Use of JWT (JSON Web Tokens)

Insecure OAuth2/OpenID Connect Implementation

Improper Certificate Validation (e.g., SSL/TLS certificate pinning)

Insecure Third-Party Library Dependencies

Subresource Integrity (SRI) Bypass

DNS Rebinding

Server Name Indication (SNI) Injection

Insecure use of HTTP Headers

Insecure use of Cookies

Insecure use of HTML5 features (e.g., Web Storage, Web Workers)

Insecure GraphQL Implementation

Insecure use of Content Delivery Networks (CDNs)

Insecure WebAssembly Implementation

Insecure Single Sign-On (SSO) Implementation

Insecure Server-Sent Events (SSE) Implementation

Improper use of HTTP Security Headers

Cache-Control Misconfiguration

Improper Input Validation

Insecure Direct Memory Access (DMA)

Insecure Integration with Third-Party Services

Cross-Origin Opener Policy (COOP) Bypass

Cross-Origin Embedder Policy (COEP) Bypass

Insecure usage of Cloud Storage Services (e.g., AWS S3, Azure Blob Storage)

Insecure Server Configuration (e.g., Default Credentials, Unpatched Systems)

Exposed Administration Interfaces

Insecure use of Serverless Architectures (e.g., AWS Lambda, Azure Functions)

Insecure use of Microservices

Insecure use of Content Management Systems (CMS) and Plugins

Insecure use of gRPC (gRPC Remote Procedure Calls) Implementation

Insecure use of QUIC (Quick UDP Internet Connections) Protocol

Insecure use of HTTP/3 Protocol

Insecure use of Edge Computing Services (e.g., AWS Lambda@Edge, Cloudflare Workers)

Insecure use of Server Push Technology (e.g., HTTP/2 Push)

Insecure use of Progressive Web Apps (PWAs)

Insecure use of WebRTC (Web Real-Time Communication) Implementation

Insecure use of Service Workers

Insecure use of Message Queues (e.g., RabbitMQ, Apache Kafka)

Insecure use of Browser Extensions and Add-ons

Insecure use of Third-Party APIs and SDKs

Insecure use of Machine Learning Models and APIs

Insecure use of AI-based Services

Insecure use of IoT Devices and APIs

Insecure use of Container Orchestration Platforms (e.g., Kubernetes, Docker Swarm)

Insecure use of Server-Side Rendering (SSR) Frameworks (e.g., Next.js, Nuxt.js)

Insecure use of Client-Side Rendering (CSR) Frameworks (e.g., React, Angular, Vue)

Insecure use of WebSockets over TLS (WSS)

Insecure use of HTTP Alternative Services

Insecure use of HTTP Strict Transport Security (HSTS) Preloading

Insecure use of Web Cryptography API

Insecure use of Web Authentication API (WebAuthn)

Insecure use of Custom Protocol Handlers

Insecure use of Web Payment API

Bug Hunting Checklist

Strategy
Tips

- First of all, get to know your application, spend some time on using it normally but keep burp open
- Start with XSS and SSTI as early as possible if in scope, insert them into every field you see
- Create your own fuzzing lists
- Expand this list with your own findings
- VDP over paid if you want less competition
- PoC or GTFO
- Prove your impact

Sessions
- Make a user with every role and check if he can directly access pages he should not be able to
- Take away a role and check if the user can still do the actions before logging out
- Look at the session token, does it change? If not, they might be useable for session fixation
- Delete a logged in user and check if he can still do actions before logging out

Stored XSS
Tips
- For every input field
    - Try to get test an entity in
    - Try to get an obfuscated entity in
    - If it catches on anything, go deeper

video's
- https://www.youtube.com/watch?v=uHy1x1NkwRU

Reflected XSS
Tips
- Check the error pages (404,403,..) sometimes they contain reflected values
    - Trigger a 403 by trying to get the .htaccess file
- Try every reflected parameter

video's
- https://www.youtube.com/watch?v=wuyAY3vvd9s
- https://www.youtube.com/watch?v=GsyOuQBG2yM
- https://www.youtube.com/watch?v=5L_14F-uNGk
- https://www.youtube.com/watch?v=N3HfF6_3k94

DOM XSS
Tips
- Would not recommend manually looking for DOM XSS
- Burp suite PRO scanner can find DOM XSS
- Tool: https://github.com/dpnishant/ra2-dom-xss-scanner

video's
- https://www.youtube.com/watch?v=gBqzzhgHoYg
- https://www.youtube.com/watch?v=WclmtS8Ftc4

Blind XSS
Tips
- Use xsshunter
- Start hunting for blind XSS soon
- Copy every payload from your xsshunter payloads section and paste it into every field you see
- XSS hunter contains a payload for CSP bypass
- Generate some variations of your payloads (example replace < with <)

XSS filter evasion tips
Tips
- < and > can be replace with html entities < and >
- You can try an XSS polyglot
    - javascript:/*--></title></style></textarea></script></xmp><svg/onload='+/"/+/onmouseover=1/+/[*/[]/+alert(1)//'>
    - https://gist.github.com/michenriksen/d729cd67736d750b3551876bbedbe626

Command injection
Tips
- Create your own fuzzing list! (See video's)
- Blind command injection can happen so make sure you include a delay command in your fuzzing list
- Make sure you include windows and Linux commands on your fuzzing list

Video's
- https://youtu.be/GZyoEXdezZM
- https://youtu.be/B-aVRsaQTgo

CSRF
Tips
- Check if there is a CSRF token
- Check if the token changes
- Check if the server still accepts the token if you give it a random token

video's
- https://www.youtube.com/watch?v=ImqLIFMQrwQ

Cookie
Tips
- Httponly flag?
- Secure flag?
- Is the domain of the cookie checked?
    - If not You can write a cookie to a subpath and it will append that to the request
- Is cookie reflected in URL GET parameter?

IDOR/broken access control
Tips
- Try directly going to objects that you have no right to that are on the same level of authentication as the user
- Try directly going to objects that you have no right to that are on a higher level of authentication as the user
- Try directly going to objects that you have no right to that are from a different client in the system
- In Europe, Personal identifiable information is sensitive so bugs containing these can be of a higher severity

video's

- https://www.youtube.com/watch?v=hZ0xSRswN8M
- https://www.youtube.com/watch?v=mjrGOuFc1Kw
- https://www.youtube.com/watch?v=5vYhTik8_yU
- https://www.youtube.com/watch?v=HQUxXE1oaIE

LFI/RFI

Tips

- If a file or image is being loaded from the local disk, try LFI/RFI (example file=test.jpg)
- Keep a VPS handy for when you need to do RFI

SSTI

Tips

- Try to inject }}{{7*7}}
- Try to inject }}[[7*7]]

video's

- https://www.youtube.com/watch?v=iQ6FuL-DgX8
- https://www.youtube.com/watch?v=i8cvh0u-VXE

XXE

Tips

- For every XML input you see, try XXE
- For every document upload, try to upload a docx file. Those can be vulnerable to XXE as well.
- For every picture upload, try to upload an SVG, you can do XXE via SVG as well

video's

- https://www.youtube.com/watch?v=AQUHzyzZXeA
- https://www.youtube.com/watch?v=unS3xGZj8xk
- https://www.youtube.com/watch?v=EwsW2WfUTmQ

SSRF

Video's

- https://youtu.be/UpzXZcfYNNc
- https://youtu.be/unS3xGZj8xk

Chaining XSS

Tips

- Maybe use XSS to steal non httpOnly cookie?
- Maybe use XSS to overwrite cookie on different path?
- Maybe use session that never changes togheter with xss to steal cookie for eternal account takeover (+ severity)
- Maybe use XSS to steal info displayed on the page (GDPR issue - PILL data)

Videos

- https://www.youtube.com/watch?v=5vYhTik8_yU
- https://www.youtube.com/watch?v=unS3xGZj8xk

Chaining CSRF

Tips

- Maybe use a CSRF to make someone insert XSS on their own page?

Chaining IDOR

Tips

- Sometimes, programs use UUID's (1213804129abc80823213214) and you find an IDOR on it. It will be low impact because you can't easily guess this id. maybe you can find an endpoint that displays all the UUID's?

Finding hidden endpoints

Tips

- Read the documenation. If there's an API, there might be API docs, google them!
- If there is a mobile app but it's not in scope, the app might communicate with a server that is in scope so think outside of the box
- GAU is a great tool to analyse JavaScript files
- Look in the settings if you can find some modules that are not active by default. Every hurdle you take is one that leaves a few other hackers behind
- If there is a paywall, invest. The more you have access to, the better

Web App Pentesting Checklist

CSRF:
- Check if the token is present on any form it should be
— ONLY Create, Update and Delete forms should have CSRF tokens
- Server checks if the token length is correct
- Server checks if parameter is there
- Server accepts empty parameter
- Server accepts responds without CSRF token
- Token is not session bound

JWT:
- None-signing algorithm is allowed
- Secret is leaked somewhere
- Server never checks secret
- Secret is easily guessable or brute-forceable

Open redirect bypass:
- evil.com/expected.com
- Javascript openRedirects
- Hidden link open redirects
- Using // to bypass
- https:evil.com (browser might correct this, filter might not catch it)
- /\ to bypass
- %00 to bypass (null byte)
- @ to bypass
- Parameter pollution (adding the same parameter twice)

BAC
- Test higher Priv functions should not be able to be executed by lower Priv user
—Test ALL user levels
— Test with authorise
— JS Functions via developer console
— Copy and paste of URL

IDOR
- Test between ALL tenants (companies hosted on one server/database. Can also be divisions of companies)
— Test with authorise
— JS Functions via developer console
— Copy and paste of URL

Captcha bypasses
- Try change request method
- Remove the captcha param from the request
- leave param empty
- Fill in random value

LFI
- Using // to bypass
- /\ to bypass
- \\

- %00 to bypass (null byte)
- @ to bypass
- URL encoding
- double encodings

RFI:
- Using // to bypass
- /\ to bypass
- \\
- %00 to bypass (null byte)
- @ to bypass
- URL encoding
- double encodings

SQLi:
- '" to trigger
— SQLmap

XXE:
- SVG files (images), DOCX/XLSX, SOAP, anything XML that renders
- Blind SSRF, file exfiltration, command exec

Template injections (CSTI/SST)
- ${7*7}
- If resolves, what templating engine
- Try exploit by looking at manuals
— URL encode special chars ({}*)
— HTML entities
— Double encodings

XSS:
- '"`><img src=x> into every input field, the moment you register and start using the application
- Enter a random value into every parameter and look for reflection
- See what context reflection is in
- Craft attack vector based on context
— JS
— HTML
— HTML tag attribute
— ...
— Url encode
— HTML entities
— Capital letters
— BASE64 encode payload
- CSP might be active
— Try bypasses
— See what is active and where script can be gotten from
— Encode them in base64
— Mascarade script as data

SSRF
- SSRF against server itself
- SSRF against other servers on the network

Command injection
- Test every single parameter
- Make a list of commands + command separators for target OS

Admin panel bypass
- Try referr header
- Easy username/pass
- Directory brute forcing for unprotected pages

Web Application Security Testing Methodology

1. Information gathering

- Search engine recon
  - Google dorking/yahoo dorking/...
    - Site:target.com -www -xxx...
    - We work exclusively, not inclusively. This means we start at the base site:target.com and exclude results we viewed. This ensures we don't miss things. We ALSO work inclusively, but it's only after working exclusively
- Webserver fingerprint
- Source code investigations
- Asset discovery
- User emulation (Keep MiTM proxy open for site map population)
- Endpoint discovery
- Enumerate any admin interfaces
- Enumeration of errors and stack traces
- Repository recon
  - github dorking
- Fingerprint the application
- Map the application architecture
- Map out integration points
- Find
  - Data flows
  - Paths
  - Race's

2. Exploits

Login system

- Test the JWT token
  - Signature Verification
    - The none signing algorithm sometimes is accepted
    - Weak HMAC Keys
    - HMAC vs Public Key Confusion
    - https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/10-Testing_JSON_Web_Tokens
- Login bypass
  - Directory brute force
  - SQLi
  - Session ID prediction
- Username enumeration
- Credentials transported over HTTP
- Default credentials
- Issues in the registration process
  - Tokens sent over plaintext?
  - DoS by entering too many characters
  - Register a user with XSS attack vector in every input field, use for further testing
- Weak password systems
- Test password reset systems
  - Add second email parameter with email of attacker
  - Any tokens sent over HTTP
  - Weak predictable tokens

- Is the user forced to reauthenticate
- SessionID in URL
- Logout should invalidate session tokens
- Validate that a hard session timeout exists.
- Weak lockout
- Is there any alternative login system
  - Oauth
  - Mobile login
  - Token login with weak tokens
- Testing for session puzzling
  - https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/08-Testing_for_Session_Puzzling
- Session hijacking
  - Request from attackers website to victims bank for example to login. The bank will possibly return session vars if bad config. This leads to attackers owning session vars now.
  - https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/09-Testing_for_Session_Hijacking

3. Input validation

- XSS
  - Reflected XSS
  - Stored XSS
  - Blind XSS
  - DOM XSS
- SQLi
- XXE
- Command injection
- SSTI/CSTI
  - Insert ${7*7} into every field you see, if it resolves, investigate further
- SSRF
  - Add burp collaborator URL in everywhere that URL resolves
- HTTP parameter polution
  - https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/04-Testing_for_HTTP_Parameter_Pollution
- LDAP injection
- SSI (server side includes injection)
- XPath injection
- Clickjacking
- GraphQL testing

4. General exploits

- RFI/LFI
- Insecure session management
- Subdomain takeover
- Check if user role is user controllable (Can you make yourself admin)
- Test the cookies attributes
  - Sensitive cookies should have secure and httponly flag
  - Domain and path need to be set right
  - Expires in timely manner
  - SameSite Attribute
- CSRF testing
  - Only on sensitive functions, not on login/logout
- File upload testing
  - Uploading of malicious content
  - File upload restrictions
    - Changing mimetype
    - Using nullbytes
- Test integration points for overextended priviledges
- Test if the browser caches sensitive information
  - Use the back button after logout timer
  - Click around after login timer
  - Check cache headers on sensitive pages
- There should not be any weak encryption used anywhere
  - Search for the following keywords to identify use of weak algorithms: MD4, MD5, RC4, RC2, DES, Blowfish, SHA-1, ECB

5. Business logic flaws

- Data validation
  - example may be if I use my credit card at multiple locations very quickly it may be possible to exceed my limit if the systems are basing decisions on last night's data.
  - Identify data injection points.
  - Validate that all checks are occurring on the back end and can't be bypassed.
  - Attempt to break the format of the expected data and analyze how the application is handling it.
- Test for hidden parameters that you can change with impact.
  - For example changing account type from consumer to business
- Check for things that are only hidden in the front-end
- Check for disabled fields that are only front-end disabled
- Integrity checks
  - Review the project documentation for components of the system that move, store, or handle data.
  - Determine what type of data is logically acceptable by the component and what types the system should guard against.
  - Determine who should be allowed to modify or read that data in each component.
  - Attempt to insert, update, or delete data values used by each component that should not be allowed per the business logic workflow.
- Test functions that can only be used a limited amount of times
  - For example a coupon code that you should only be applying one time but that's just a front-end check
- If something gets added to account and should be withdrawn again, check if it is.
  - For example if you order an item but cancel the order, your loyatee points should go down as well.

Bug Bounty

Authentication Bypass
NO RATE LIMIT
XSS
CSRF
CORS
SSRF
SQL Injection
Clickjacking
Broken Link Hijack
Subdomain Takeovers
HTML Injection
Local File Inclusion (LFI)
Directory Listing
RCE
Broken Authentication
Broken Object Level Authorization
IDOR
Insecure Deserialization
JWT Auth Bypass
Cache Poisoning
Open Redirect
Sensitive Data Exposure
Server Side Request Smuggling
Session Hijacking
Server Side Template Injection
Web Application Firewall (WAF) Bypass
XML Entity Expansion (XEE)
XML External Entity (XXE)
DOM XSS (DOM-Based Cross-Site Scripting)
Insecure OS/Firmware
Insufficient Transport Layer Protection
LDAP Injection
Open Redirect DOM Based Attacks
OS Command Injection
Reflected XSS
Security Misconfiguration
Stored XSS
Cryptography Failure
Cleartext Transmission Of Session Token
Content Spoofing
Session Fixation
OAuth Misconfiguration

BugBounty (Hunter)

- Payloads
- JSON
- RCE
  - PHPMailer
  - Node.js
- One liner bugbounty
- CMS exploitation
  - Wordpress
- Basic enumeration
  - find subdomains
  - brute force > - hydra - ffuf
- Polyglot payloads
  - XSS polyglot
  - SQLi polyglot
  - JS/URL polyglot
- Cookie hacking
- Bypass technics
  - Bypass file upload filtering
  - Bypass 401/403
  - Bypass 429
  - Bypass password reset
  - Bypass LFI WAF
- Open redirect
- Cross Site Scripting (XSS)
  - Common payloads
  - XSS to LFI
  - Top XSS dorks
- XML External Entity (XXE)
  - Exploitable Protocols
- Server Side Template Injection (SSTI)
  - SSTI CheatSheet
- Sever Side Request Forgery (SSRF)
- Client Side Request Forgerty (CSRF)
- Carriage Return and Line Feed (CRLF)
- Local File Inclusion (LFI)
- Remote File Inclusion (RFI)
- Structured Query Language injection (SQLi)
- Insecure deserialization

Linux

- Basic enumeration
- cURL Cheat Sheet
- ZIP all in one
- Useful find commands
- Keyboard shortcuts for terminal
- Simple bash port scanner
- Python virtual environment
- Specific user file permission
- SMB enumeration
- Redis

Windows

- Basic enumeration
- SMB enumeration
- MS SQL
- Xfreerdp

Linux Privesc

- LXC/LXD container
- Perl setuid capability

Windows privesc

- metasploit
  - local_exploit_suggester
  - mimikatz_kiwi
- mimikatz
- impacket
  - psexec
  - smbexec
  - wmiexec
  - dcomexec
  - crackmapexec
  - smbclient
- evil-winrm

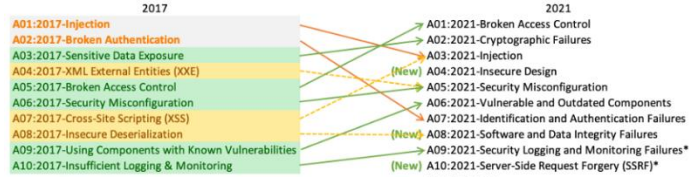
Android
- ADB Basics

Extra notes
- make NTML hash from password
- snap
- SMTP
- SQLi+XSS+SSTI
- ShellShock
- grep
  - emails
  - urls
- Password-List

Vulnerabilities Addressed During Audit

✓ **OWASP Top 10**

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]



A1 – Injection
- SQL injection
- LDAP injection
- OS commanding
- SSI injection
- X path injection

A2 – Broken Authentication
- Session management
- Privilege escalation
- Insufficient session expiration

A3 – Sensitive Data Exposure
- Test for sensitive data exposure
- Testing for critical data management

A4 - XML External Entities (XXE)
- Testing for XML external entities

A5 - Broken Access Control
- Testing for unauthorized functionality
- Insecure data object reference

A6 - Security Misconfiguration
- Insecure cryptographic
- storage, Insufficient transport layer protection,
- Check for SSL certificate attributes
- Misconfiguration of OS, libraries, frameworks etc.

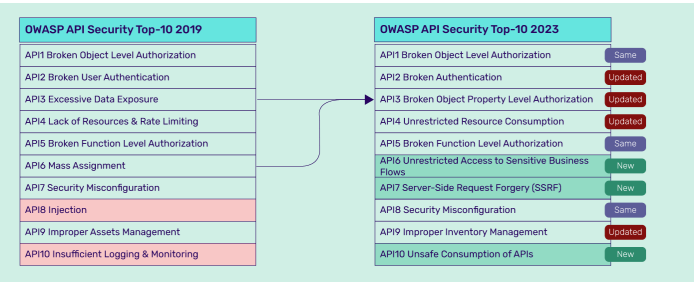
A7 - Cross-Site Scripting (XSS)
- Testing for cross-site scripting

A8 - Insecure Deserialization
- Testing for Insecure deserialization

A9 - Using Components with Known Vulnerabilities
- Known vulnerable framework and Library
- Check for vulnerable software modules, product CVEs

A10 - Insufficient Logging & Monitoring
- Secure Communication, API authentication, Data formats,
- Access control, Injection attack in API,
- Insufficient logging and monitoring Logical Checks
- Abuse of Functionality
- Insufficient Anti-automation
- Insufficient Authentication
- Email ids can be harvested for spamming
- ByPass Authentication
- Insufficient password recovery
- Insufficient process validation
- Application does not display last login time
- Server-side validation

✓ **OWASP API Top 10 Risks**



✓ OWASP Mobile Top 10 2024

- M1: Improper Credential Usage
- M2: Inadequate Supply Chain Security
- M3: Insecure Authentication/Authorization
- M4: Insufficient Input/Output Validation
- M5: Insecure Communication
- M6: Inadequate Privacy Controls
- M7: Insufficient Binary Protections
- M8: Security Misconfiguration
- M9: Insecure Data Storage
- M10: Insufficient Cryptography

Comparison between 2016 and 2024

Comparison Between 2016-2024		
OWASP-2016	OWASP-2024-Release	Comparison Between 2016-2024
M1: Improper Platform Usage	M1: Improper Credential Usage	New
M2: Insecure Data Storage	M2: Inadequate Supply Chain Security	New
M3: Insecure Communication	M3: Insecure Authentication / Authorization	Merged M4&M6 to M3
M4: Insecure Authentication	M4: Insufficient Input/Output Validation	New
M5: Insufficient Cryptography	M5: Insecure Communication	Moved from M3 to M5
M6: Insecure Authorization	M6: Inadequate Privacy Controls	New
M7: Client Code Quality	M7: Insufficient Binary Protections	Merged M8&M9 to M7
M8: Code Tampering	M8: Security Misconfiguration	Rewording [M10]
M9: Reverse Engineering	M9: Insecure Data Storage	Moved from M2 to M9
M10: Extraneous Functionality	M10: Insufficient Cryptography	Moved from M5 to M10

Vulnerabilities that didn't make the place on the initial release list, but in the future, OWASP may consider them.

- Data Leakage
- Hardcoded Secrets
- Insecure Access Control
- Path Overwrite and Path Traversal
- Unprotected Endpoints (Deeplink, Activitity, Service …)
- Unsafe Sharing

Top 10 Mobile Risks - Final List 2016

- M1: Improper Platform Usage
- M2: Insecure Data Storage
- M3: Insecure Communication
- M4: Insecure Authentication
- M5: Insufficient Cryptography
- M6: Insecure Authorization
- M7: Client Code Quality
- M8: Code Tampering
- M9: Reverse Engineering
- M10: Extraneous Functionality

Top 10 Mobile Risks - Final List 2014

- M1: Weak Server Side Controls
- M2: Insecure Data Storage
- M3: Insufficient Transport Layer Protection
- M4: Unintended Data Leakage
- M5: Poor Authorization and Authentication
- M6: Broken Cryptography
- M7: Client Side Injection
- M8: Security Decisions Via Untrusted Inputs
- M9: Improper Session Handling
- M10: Lack of Binary Protections

OWASP Cloud-Native Top 10

CNAS-1: Insecure cloud, container or orchestration configuration
Examples:

- Publicly open cloud storage buckets
- Imrpoper permissions set on cloud storage buckets
- Container runs as root
- Container shares resources with the host (network interface, etc.)
- Insecure Infrastructure-as-Code (IaC) configuration

CNAS-2: Injection flaws (app layer, cloud events, cloud services)
Examples:

- SQL injection
- XXE
- NoSQL injection
- OS command injection
- Serverless event data injection

CNAS-3: Improper authentication & authorization
Examples:

- Unauthenticated API access on a microservice
- Over-permissive cloud IAM role
- Lack of orchestrator node trust rules (e.g. unauthorized hosts joining the cluster)
- Unauthenticated orchestrator console access
- Unauthrized or overly-permissive orchestrator access

CNAS-4: CI/CD pipeline & software supply chain flaws
Examples:

- Insufficient authentication on CI/CD pipeline systems
- Use of untrusted images
- Use of stale images
- Insecure communication channels to registries
- Overly-permissive registry access
- Using a single environment to run CI/CD tasks for projects requiring different levels of security

CNAS-5: Insecure secrets storage
Examples:

- Orchestrator secrets stored unencrypted
- API keys or passwords stored unencrypted inside containers
- Hardcoded application secrets
- Poorly encrypted secrets (e.g. use of obsolete encryption methods, use of encoding instead of encryption, etc.)
- Mounting of storage containing sensitive information

CNAS-6: Over-permissive or insecure network policies
Examples:

- Over-permissive pod to pod communication allowed
- Internal microservices exposed to the public Internet
- No network segmentation defined
- End-to-end communications not encrypted
- Network traffic to unknown or potentially malicious domains not monitored and blocked

CNAS-7: Using components with known vulnerabilities
Examples:

- Vulnerable 3rd party open source packages
- Vulnerable versions of application components
- Use of known vulnerable container images

CNAS-8: Improper assets management
Examples:

- Undocumented microservices & APIs
- Obsolete & unmanaged cloud resources

CNAS-9: Inadequate 'compute' resource quota limits
Examples:

- Resource-unbound containers
- Over-permissive request quota set on APIs

CNAS-10: Ineffective logging & monitoring (e.g. runtime activity)
Examples:

- No container or host process activity monitoring
- No network communications monitoring among microservices
- No resource consumption monitoring to ensure availability of critical resources
- Lack of monitoring on orchestration configuration propagation and stale configs

SANS TOP 25

- SQL Injection
- OS Command Injection
- Cross-Site Scripting
- Malicious File Upload
- Uncontrolled Format String
- Integer Overflow or Wraparound
- Missing Authentication for Critical Function
- Missing Authorization
- Cross-Site Request Forgery (CSRF)
- Buffer Overflow
- Path Traversal
- Download of code without Integrity Check
- Inclusion of Functionality from Untrusted Control Sphere
- Use of Potentially Dangerous Function
- Incorrect Calculation of Buffer Size
- Use of Hard-coded Credentials
- Missing Encryption of Sensitive Data
- Reliance on Untrusted Inputs in a Security Decision
- Execution with Unnecessary
- Privileges
- Incorrect Authorization
- Incorrect Permission
- Assignment
- for Critical Resource
- Use of a Broken or Risky
- Cryptographic
- Algorithm
- Improper Restriction of
- Excessive
- Authentication Attempts
- Use of a One-Way Hash
- without a Salt
- Indusface Defined
- Checks
- Check for Encoding
- Password Auto-Complete
- Improper implementation of
- SSL (cipher, version)
- Service enumeration
- Port scanning
- Hidden iframe detection
- Malicious file can be uploaded on the server
- Steal password from browser memory
- Check HTTP methods
- Check for cookie attributes

MITRE

2023 MITRE & SANS CWE Top 25 Most Dangerous Software Weaknesses

- Out-of-bounds Write
- Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
- Use After Free
- Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
- Improper Input Validation
- Out-of-bounds Read
- Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- Cross-Site Request Forgery (CSRF)
- Unrestricted Upload of File with Dangerous Type
- Missing Authorization
- NULL Pointer Dereference
- Improper Authentication
- Integer Overflow or Wraparound
- Deserialization of Untrusted Data
- Improper Neutralization of Special Elements used in a Command ('Command Injection')
- Improper Restriction of Operations within the Bounds of a Memory Buffer
- Use of Hard-coded Credentials
- Server-Side Request Forgery (SSRF)
- Missing Authentication for Critical Function
- Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')
- Improper Privilege Management
- Improper Control of Generation of Code ('Code Injection')
- Incorrect Authorization
- Incorrect Default Permissions
  - For more HELP

2023 MITRE & SANS CWE "On the Cusp" – Other Dangerous Software Weaknesses

- Reachable Assertion
- Uncontrolled Search Path Element
- Improper Restriction of XML External Entity Reference
- Allocation of Resources Without Limits or Throttling
- Exposure of Sensitive Information to an Unauthorized Actor
- Incorrect Permission Assignment for Critical Resource
- URL Redirection to Untrusted Site ('Open Redirect')
- Improperly Controlled Modification of Object Prototype Attributes ('Prototype Pollution')
- Improper Certificate Validation

- Insufficiently Protected Credentials
- Missing Release of Memory after Effective Lifetime
- Uncontrolled Resource Consumption
- Authorization Bypass Through User-Controlled Key
- Improper Link Resolution Before File Access ('Link Following')
- Exposure of Resource to Wrong Sphere

The 2021 MITRE & SANSCWE Most Important Hardware Weaknesses

- CWE-1189 Improper Isolation of Shared Resources on System-on-a-Chip (SoC)
- CWE-1191 On-Chip Debug and Test Interface With Improper Access Control
- CWE-1231 Improper Prevention of Lock Bit Modification
- CWE-1233 Security-Sensitive Hardware Controls with Missing Lock Bit Protection
- CWE-1240 Use of a Cryptographic Primitive with a Risky Implementation
- CWE-1244 Internal Asset Exposed to Unsafe Debug Access Level or State
- CWE-1256 Improper Restriction of Software Interfaces to Hardware Features
- CWE-1260 Improper Handling of Overlap Between Protected Memory Ranges
- CWE-1272 Sensitive Information Uncleared Before Debug/Power State Transition
- CWE-1274 Improper Access Control for Volatile Memory Containing Boot Code
- CWE-1277 Firmware Not Updateable
- CWE-1300 Improper Protection of Physical Side Channels

The 2021 MITRE & SANS CWE Hardware Weaknesses on the Cusp

- CWE-226 Sensitive Information in Resource Not Removed Before Reuse
- CWE-1247 Improper Protection Against Voltage and Clock Glitches
- CWE-1262 Improper Access Control for Register Interface
- CWE-1331 Improper Isolation of Shared Resources in Network On Chip (NoC)
- CWE-1332 Improper Handling of Faults that Lead to Instruction Skips

Indicators of Compromise

Others

✓ Vulnerabilities

- Remote code execution (RCE)
- OS command injection
- Server-side request forgery (SSRF)
- SQL injection (SQLi)
- Local file inclusion (LFI)
- Remote file inclusion (RFI)
- Directory traversal
- Cross-site scripting (XSS)
  - Stored/persistent cross-site scripting
  - Reflected/non-persistent cross-site scripting
  - DOM-based cross-site scripting
  - Blind cross-site scripting
- Cross-site request forgery (CSRF)
- XML external entity (XXE)
- CRLF injection
- Email injection
- HTML injection
- NoSQL injection
- JSON Injection
- Insecure direct object references (IDOR)
- Unvalidated redirects and forwards
- Directory listing
- Password reset poisoning
- Buffer overflow
- Integer overflow

✓ Attacks

- Session hijacking
- Session fixation
- Session prediction
- Cookie hijacking
- Cookie poisoning
- Clickjacking
- Forced browsing
- XSS filter evasion
- Source code disclosure
- Host header attacks
- Web cache poisoning
- Privilege escalation
- Web shell
- Reverse shell
- Cross-frame scripting (XFS)
- Broken link hijacking
- SEO poisoning
- Low Orbit Ion Cannon (LOIC)
- High Orbit Ion Cannon (HOIC)
- Slowloris attack
- R.U.D.Y. attack
- Man-in-the-middle attacks (MITM)

✓ Protection

- Same-origin policy (SOP)
- HTTP Strict Transport Security (HSTS)
- Cookie security flags

✓ Tools

- Dynamic application security testing (DAST)
- Static application security testing (SAST)
- Interactive application security testing (IAST)
- Software composition analysis (SCA)
- Web application firewall (WAF)
- Web asset discovery
- Vulnerability management
- Vulnerability assessment

- Buffer Overflow via Environment Variables
- Server Side Include (SSI) Injection
- Cross Zone Scripting
- Command Line Execution through SQL Injection
- Object Relational Mapping Injection
- SQL Injection through SOAP Parameter Tampering
- Double Encoding
- Subverting Environment Variable Values
- Format String Injection
- LDAP Injection

- Client-side Injection-induced Buffer Overflow
- Input Data Manipulation
- Flash Injection
- XSS Using MIME Type Mismatch
- Exploiting Trust in Client
- File Content Injection
- Serialized Data with Nested Payloads
- Oversized Serialized Data Payloads
- Filter Failure through Buffer Overflow
- XML Injection
- Fuzzing for garnering other adjacent user/sensitive data
- Leverage Alternate Encoding
- Fuzzing
- Using Leading 'Ghost' Character Sequences to Bypass
- Input Filters
- Accessing/Intercepting/Modifying HTTP Cookies
- MIME Conversion
- Exploiting Multiple Input Interpretation Layers
- Buffer Overflow via Symbolic Links
- Overflow Variables and Tags
- Buffer Overflow via Parameter Expansion
- Signature Spoof
- Embedding NULL Bytes
- Postfix, Null Terminate, and Backslash
- DOM-Based XSS
- Cross-Site Scripting (XSS)
- Using Slashes and URL Encoding Combined to Bypass
- Validation Logic
- Server-Side Request Forgery
- String Format Overflow in syslog()
- Blind SQL Injection
- Using Unicode Encoding to Bypass Validation Logic
- URL Encoding
- User-Controlled Filename
- Using Escaped Slashes in Alternate Encoding
- Using Slashes in Alternate Encoding
- Buffer Overflow in an API Call
- Using UTF-8 Encoding to Bypass Validation Logic
- Web Server Logs Tampering
- XPath Injection
- AJAX Footprinting
- OS Command Injection
- Buffer Overflow in Local Command-Line Utilities

List of Popular IT Security Standards

The good news for IT security professionals is that there are a number of IT security standards that they can use as a guideline when developing or implementing IT projects. Remember that these standards are well thought out and proven practices that can improve information security goals of your organization. Only the popular and globally recognized IT security standards have been presented in this post.

- **BITS Financial Services Roundtable (www.bits.org/FISAP/index.php)**: this is a set of Security assessment questionnaire and review process which has been developed using ISO/IEC 27002. (Also information on the overlaps between ISO/IEC 27002, PCI-DSS 1.1 and COBIT.
- **Common Criteria (www.commoncriteriaportal.org/thecc.html):** it does not provide any standards. Instead, it will give you a common set of Provides Criteria to evaluate your IT security status. These criteria also have been published as ISO/IEC 15408.

- **ISO/IEC 27001:2013:** this specifies the standards of information security management system, which consists of ten short clauses and a long annex. If your organization follows these standards, you can apply for certification to an accredited certification body. But before applying for certification, you need to go through a formal audit process.
- **NIST special publication 800-171 series:** this is basically a computer security report that addresses general guidelines and research outcomes on computer security, conducted by academics, industries and governments.
- **ISO27002:2013:** this is an information security standard developed by ISO from BS7799 (British standard of information security). This standard describes general controls of IS security, which is helpful for those who both implement and manage information systems.

- **COBIT 5**-it stands for Control Objectives for Information and Related Technology, which was developed by ISACA for IT governance and management. One of the important parts of COBIT is to provide a set of controls to mitigate IT risk. To complement COBIT, you can use RISK IT framework, also developed by ISACA, in order to manage all types of risks related to the use of IT.

Note:If you want to learn about ISO standards in simple English you can use the following link that explains a list of useful information systems management standards.
http://praxiom.com/#ISO_IEC_27001_2005_LIBRARY_

For those who wants to explore more specific ISO standards for information security can have a look at **ISO/IEC 27000-series** , which is a family of IS management standards. Even if you do not want to spend money on ISO certification or any other accreditation, you can follow these standards in order to enhance the overall security of your IT and relevant assets.

63 Web Application Security Checklist for IT Security Auditors and Developers

As you know that every web application becomes vulnerable when they are exposed to the Internet. Fortunately, there are a number of best practices and coutner measures that web developers can utilize when they build their apps. This post will list some proven counter measures that enhance web apps security significantly.

Network security checklist

1. Most of the web applications reside behind perimeter firewalls, routers and various types of filtering devices. Always make sure that your perimeter devices used for filtering traffic are stateful packet inspection device.
2. Routers and firewalls should be configured to allow necessary types of traffic such as http or https. Block all other unnecessary types of traffic that you do not need to support your web applications.
3. Just like inbound traffic you need to allow outbound traffic. Configure your router and firewall for the necessary outbound traffic from your web applications.
4. Make sure your perimeter devices (firewall, routers etc.) are equipped with appropriate DOS (denial of service) countermeasures.

If you are using Cisco routers, you can use rate-limit commands in order to limit the committed access rate.

5. If you are using load balancers, check out whether it is disclosing any information about your internal networks.
6. Think about implementing a network intrusion system and establish appropriate policies and procedures to review logs for attack signature.
7. Disable telnet access to all of your network devices for remote access. Use SSH for only for the devices that you need to access for the Internet.
8. Make a password change policy for all of your remote access devices and also allow only specific IP addresses to access your network remotely.
9. Conduct network vulnerability scans regularly.
10. Every time you make major changes to your network, you may arrange for a penetration test by a third party organization. Make a plan to conduct penetration test at least each year.

Web Server checklist

1. Whenever your software vendor release software updates or any security patches, apply it to your network after appropriate testing.
2. Check your server configuration to ensure that it is not disclosing any sensitive information about the install application software in your server.
3. Disallow servers to show directory listing and parent path.
4. Disable the unnecessary services on your servers.
5. If your software vendor recommends you to use specific security settings, implement it appropriately.
6. Disable or delete guest accounts, unnecessary groups and users.
7. Enable OS auditing system and web server logging.
8. Remove unnecessary modules or extension from your web servers.
9. Remove default website and sample contents, if there is any, from all of your web servers.
10. Configure authentication mechanism properly in your server directories.
11. Always use SSL when you think your traffic is sensitive and vulnerable to eavesdroppers. Make sure you use the appropriate key length for encryption ad use only SSLv3.
12. Deploy web contents in a virtual root that do not have any administrative utilities. This virtual root can be a separate drive or separate disk.
13. Disable directory listing and parent path in your web server.
14. Check your current error message pages in your server. If it is leaking any information about your server, customize it.
15. Make sure all the accounts running HTTP service do not have high level privileged.
16. Create access control list for all of your web directories and files.
17. If your servers have WebDAV (Web Distributed Authoring and Versioning) disable it or delete it if you do not need it. If you have to keep WebDAV, apply proper access restrictions to it.
18. Disable web publishing functionalities (such as iPlanet products) if you have any.
19. Apply and fine tune your web servers security modules(UrlSCAN in IIS or Mod-security in Apache)
20. Scan your server with popular scanners in order to identify vulnerabilities and mitigate the risks.
21. Think about using host based intrusion detection system along with network intrusion system. Make a policy to review the logs.

Database Server security checklist

1. Check that if your database is running with the least possible privilege for the services it delivers.
2. Update your database software with latest and appropriate patches from your vendor.
3. Remove all sample and guest accounts from your database.
4. The dynamic sites need to communicate with the database server to generate request contents by the users. Restrict traffic FLOW between database and web server using IP packet filtering.
5. Use appropriate authentication mechanism between your web servers and database servers.
6. If your database has a default account, you can either change it or use a separate password.
7. Make sure database users are granted privileges according to their roles and requirements.
8. Delete extended stored procedures and relevant libraries from our database if you do not need them.
9. Do not embed database user passwords in the application codes.
10. Plan for a database audit.
11. Change database passwords after predefined period. After predefined period.

Application security

1. Create a thereat model of your application and approve it by the management and IS security team.
2. Segregate the application development environment from the production environment. Never use the production data in the test environment for testing purpose.
3. Make sure your application's authentication system match industries best practices.
4. Use ACL to control access to application directories and files.
5. Use proper input validation technique output encoding in the server side.
6. Secure the source codes and files of your web applications.
7. Remove temporary files from your application servers.
8. Cookies and session management should be implemented according the best practices of your application development platform. Implement a session expiration timeout and avoid allowing multiple concurrent sessions.
9. Assign a new session ID when users login and have a logout option.
10. Allow least privilege to the application users.
11. Implement a CAPTCHA and email verification system if you allow your users to create account with your application.
12. Use appropriate encryption algorithm to meet your data security requirements.
13. Always place the 'includes' files (the files required by the server side scripts) outside the virtual root directory. Apply ACL to your include files if possible. Rename the includes files into .asp in your IIS server.
14. Identify the vulnerable API or function calls and avoid them if there is a work around for it.
15. Parameterized SQL queries to prevent SQL injection.
16. Enable error handling and security logging features.
17. Run a security audit on your source codes.
18. Perform a black box test on our application. If you do not have any penetration tester in your organization, which is more likely, you can hire a professional penetration tester.
19. Change administration and other privileged passwords regularly.
20. Conduct web application vulnerability scan regularly to identify application layer vulnerabilities of your application.
21. Always conduct a proper penetration test before moving your application from the development environment to the production environment. Also, run a pen test when you make signification modification to the application.

How to Configure VPN in Cisco Routers

Virtual private network can be configured with most of the Cisco routers(800 to 7500 series) with IOS version 12 or higher.VPN can be implemented in a number of ways–with various level of security measures and configuration. To determine the right VPN configuration for your network, you need to have a solid understanding in cryptographic system and encryption algorithm.Besides, one needs to know which type of VPN is suitable for remote clients and which type of VPN is used to create secure site-to-site connection. This article explains the necessary steps with configuration script to setup VPN in Cisco routers. This configuration can be simulated in Cisco packet tracer software as well.

Types of VPN

1. Remote access VPN
2. Site to site VPN
3. Business partner vpn

VPN implementation methods
The two methods that can be used to implement any of the above mentioned three types of VPN are:

1. IPsec based VPN
2. SSL based VPN

Both types of VPN implementation method has its advantage and disadvantages. If you choose to implement an IPsec based VPN, you need to install client software on every remote host or devices that need to access the VPN. On the other hand, SSL VPNs can directly establish connection between two machines without the need of installing any client software; it is possible because SSL basically a web browser based VPN solution.

Most of the site-to-site and business partner types VPNs are IPsec based, whereas SSL is widely used for remote client access VPNs

IPsec VPN

The main purpose of IPsec is to provide communication security while your data pass through the public network such as Internet. To establish IPsec connection, you need to have IPsec compliant devices such as Cisco IOS based routers. The following cryptographic technology is used with IPsec

- **Diffie-Hellman** key exchange
- Public key cryptography
- Data encryption algorithm-it helps to validate the identity of the sender and
- **Hashing algorithm**– it verifies authenticity and intergrity of data. Hash algorithms used are HMAC,SHA-1,MD5
- **Digital certificate**– a way to validate the identity of the sender. Digital certificate contains the identity details of a public key holder and it is issued by a CA.

VPN Design Process

When you decide to set up a VPN, you need to design a VPN implementation plan. The VPN implementation plan needs to consider the following aspects.

1. Identify the type of VPN (SSL or IPsec) you need to implement and what the computer systems or network equipments need to be protected by VPN connection.
2. Design VPN-choose the type of authentication methods, filtering and cryptographic policy
3. Testing- it is better to try to test your design in a test environment before you deploy the VPN in your organization.

4. Deployment-once you are satisfied with the test result, you can start deploying your VPN as per your design

5. Monitoring- monitor the traffic activity at the VPN end points and always check out the security warnings or updates with your VPN equipment vendors.

IPsec Protocols

IPsec protocol is basically a combination of two different protocols with two different purposes. These two protocols are collection of security protocols are: packet protocols and service protocols. There are two major packet protocols: ESP (encapsulating security payload) and Authentication Header (AH).The service protocol of IPsec is known as IKE-Internet Key Exchange.

ESP-its encrypts entire IP data portion of the packets and adds ESP header and trailer at the end of the packet.ESP provides confidentiality, authentication and integrity to a data packet.

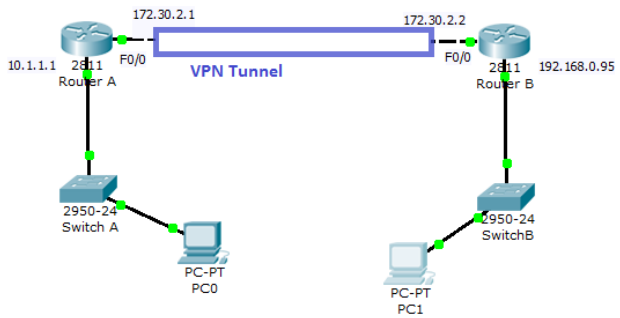
AH– authentication header adds to the IP packet to provide the data packet validation.AH does not offer any encryption service, unlike ESP.

IKE-it uses Diffie Hellman key exchange process to offer key management and security association.

So, as you see that IPsec mainly provides two type of service – packet authentication and encryption- by using ESP and AH. IPsec can provide these two services in two modes-tunnel modes and transport mode. Tunnel mode provides the encryption and authentication for the entire data packet, where as transport mode provides only the transport layer data security and authentication. Thus transport mode IPsec generates lower overhead and is faster than tunnel mode IPsec. The disadvantage of transport mode IPsec is the any attacker may perform traffic analysis of this packet since the header information is not encrypted.

Now, you understand the basics of IPsec and let's see how we can implement IPsec based VPN in a Cisco router.

This configuration is for a **site to site type VPN**, where all traffic from router A to router B will be encrypted with IPsec.



cisco

vpn configuration

Configuration on Router A

RouterA#configure terminal

RouterA(config)#crypto isakmp policy 1

RouterA(config-isakmp)#authentication pre-share

RouterA(config-isakmp)#encryption aes 128

RouterA(config-isakmp)#group 2

RouterA(config-isakmp)#exit

RouterA(config)#lifetime 96400

RouterA(config)#end

RouterA#copy run start

Now create a transform set name and give it a name as you like.For example, name the set as ciscoset

RourterA#conf term

RouterA(config)#crypto ipsec transform-set **ciscoset** esp-aes esp-sha-hmac

RouterA(cft-crypto-trans)#exit

RouterA(config)#access-list 101 permit ip 10.1.1.0 0.0.0.255 192.168.0.0 0.0.0.255

RouterA(config)#crypto map **router1torouter2** 10 ipsec-isakmp

RouterA(config-crypto-map)#set peer 172.30.2.2

RouterA(config-crypto-map)#match address 101

RouterA(config-crypto-map)#set transform-set **ciscoset**

```
RouterA(config-crypto-map)#exit
```

Next, you have to apply the crypto map to the external interface of router A

```
RouterA(config)#interface fastethernet0/0

RouterA(config-if)#crypto map route1torouter2

RouterA(config-if)#end

RouterA(config)#ip route 192.168.0.0 255.255.255.0 172.30.2.2

RouterA(config-if)#end
```

Now, you can configure the router B with similar configuration just by changing the peer IP, IP router and access list IP with for router A.

Ethical Hacker|Cybersecurity|Penetration Tester Needs-

- Citrix Desktop Breakout
- NFS, no_root_squash and SUID - Basic NFS Security
- Web Protocols
- Performing Domain Reconnaissance Using PowerShell
- Extracting Password Hashes from the Ntds.dit File
- Network Segmentation
- PowerShell for Pen Testers
- Local Linux Enumeration & Privilege Escalation Che...
- Penetration testing methodologies
- Public Vulnerability Database Resources
- Active Directory Roles
- Cryptography
- Functionality Testing a IDS/IPS
- Mobile app security testing checklist
- Unrestricted File Upload Security Testing Web Appl...
- Pre-Engagement Interactions
- Network Security VAPT Checklist
- Penetration Testing Guidelines
- CVSS for Penetration Test Results
- Active Directory Delegation Security Issues
- Kerberos Working
- Cisco Network Penetration Testing
- Cisco Router Hardening
- Mini Penetration Testing Framework
- Web Application Security Testing Cheat Sheet
- Windows Registry ACLs
- Wireless Security Protocols
- Docker Security
- Complete Domain Compromise with Golden Tickets
- PowerShell AMSI Bypass
- Network Infrastructure Penetration Testing
- Windows Privilege Escalation Scripts
- Linux Privilege Escalation Scripts
- MSSQL Database Penetration Testing
- Oracle Database Penetration Testing
- IPsec VPN Penetration Testing
- VOIP Penetration Testing Cheat Sheets
- Metasploit Cheat Sheets
- Wireless Hacking WiFu
- Applocker Bypass Technique
- Packet Crafting
- CREST CCT Application Exam
- Windows Patch Management Strategies
- Persistence
- Lateral Movement
- IPv6 for Pentesters

Ethical Hacker|Cybersecurity|Penetration Testing-

- IoT Device Penetration Testing
- Embedded Device Security Assessment
- SDR Exploitation
- ICS / OT / Security Assessment
- WEB / API / Mobile Application Security Assessment
- Secure Code Review
- Firmware Security Assessment
- Red Teaming
- Physical Pentest
- Scenario-Based Penetration Testing
- End-User Device Testing including Kiosks
- Coordination and Assistance for Third-Party Risk Advisory Certification Coverage

✓ Flagship Services-

Maritime Cyber Security Assessment

- IoT Device Penetration Testing
- Embedded Device Security Assessment
- SDR Exploitation
- ICS / OT / Security Assessment
- WEB / API / Mobile Application Security Assessment
- Secure Code Review
- Firmware Security Assessment
- Red Teaming
- Physical Pentest
- Coordination and Assistance for Third-Party Risk Advisory Certification
- Coverage

Reason

- Customer data exposure
- Corporate data exposure
- Physical damage
- High-risk downtime
- Broader liability
- Reputation and brand damage

Telecom Signaling Penetration Testing

- SS7 / SIGTRAN Penetration Testing
- Diameter Penetration Testing
- GTP Penetration Testing
- Pen-Testing GRX (GPRS Tunneling Protocol)
- Voice Over LTE (VoLTE) & Fixed IMS Pen-Testing
- Assessments of the Air Interface
- Location tracking
- MT Interception / redirection of voice calls
- MO Voice call interception
- MT SMS interception
- Denial of service
- Refuse to register a mobile device on the network.
- Retrieve a subscriber's prepaid account balance
- Use SMS/USSD to transfer prepaid balance from one account to another (if this is available in operator)
- Retrieve a subscriber's HLR profile from the home network
- Denial of data service for a subscriber / in-roamer
- IPTV Network Pen-Testing Services
- FTTH Service testing services

- IoT Device Penetration Testing

Reason

- Toll fraud
- Information harvesting & theft
- 2FA interception
- Call interception
- Location tracking
- Banking / Billing / VAS fraud
- Premium rate fraud
- Malware
- Spam
- Denial of service attacks
- Data leakage

- Embedded Device Security Assessment
- SDR Exploitation
- ICS / OT / Security Assessment
- WEB / API / Mobile Application Security Assessment
- Secure Code Review
- Firmware Security Assessment
- Red Teaming
- Physical Pentest

- Telecom Technical Security Assessment
- Telecom components configuration Security Review
- Telecom Network Elements vulnerability analysis
- Coordination and Assistance for Third-Party Risk Advisory Certification
- Coverage

Satellite Communications (SATCOM) Security Assessment

- Aeronautics
- Naval
- Armed forces/governments
- Assistance in an emergency
- Commercial (oil rigs, gas, electricity)
- Mass media
- SATCOM Infrastructure Security Review
- Component and Interface Security Assessment
- Scenario Based Penetration Testing
- Customized Red Teaming Exercise
- Embedded Device Penetration Testing
- Secure Code Review
- Firmware Analysis
- SATCOMS RF Assessment
- Critical Design Review
- Systems Integration and Integration Review

Reason

- Global positioning systems (GPS)
- Military strategies
- Data collection
- Environmental research
- Internet Operation & Integration
- Mobile phone networks
- Complex navigation systems
- Internet of Things (IoT) devices
- Large-scale electrical grids, and power suppliers.

Aviation Cyber Security Assessment

- Assessment of the Aircraft Passenger Domain
- Domain Assessment of Aircraft Information Services
- Assessment of the IFE/security IFC's
- Security Assessment of Satellite Terminals
- Assessment of Aircraft Domain Segregation
- Assessment of Gatelink wireless security (aircraft and/or airside)
- Reverse engineering of avionics hardware
- Evaluation of avionics network protocols
- Assessment of aviation radio frequency security
- Inspection of data loading / maintenance crew equipment security
- Inspection of e-enabled ground and onboard systems
- Security Review of Electronic Flight Bags (EFB)
- Customized Red Team exercise at the airport.

Reason

- Attacks on large organizations
- Critical infrastructures of various types
- Governments, and SMEs

Hotels & Hospitality Cyber Security Assessment

- Risk Advisory Compliance
- Penetration testing and independent review and assessment of hotel cybersecurity
- Conduct an audit of critical information technology assets and vulnerable points throughout your hotel's infrastructure and networks
- As new technologies and digital services are rolled out to customers, safeguard your innovation
- Examinations of physical security and social engineering
- CISO-as-a-Service (CISO-as-a-Service)
- Creation of policies and procedures
- Security Awareness training

Reason

- physical security issues (surveillance, locks, safes, guest security, etc.)
- digital assets and online privacy issues
- booking, registration, and billing issue
- systems critical for safeguarding visitors' personal and financial information

✓ Risk Advisory Services

GDPR Consulting and Audit

- GDPR Gap Analysis
- GDPR Risk Assessment
- Security Awareness Training Program
- Documentation of GDPR Rules & Regulations
- GDPR Continuation Support

coverage

- GDPR compliance Regulation

HIPAA Consultation and Audit

- HIPAA Gap Analysis
- HIPAA Risk Assessment
- Security Awareness Training Program
- Documentation of HIPAA Rules & Regulations
- HIPAA Continuation Support

Coverage

- HIPAA compliance Standard

SOC 1 Advisory Services

- SOC1 Gap Analysis
- SOC1 Risk Assessment & Treatment
- SOC1 ISMS Implementation
- Security Awareness Training Program
- SOC1 Attestation

- SOC1 Continuation Support

Coverage
- SOC1 Type 1
- SOC1 Type 2
- SSAE 18 Audit Standard

Reason
- Increased client trust, resulting in increased client retention and acquisition.
- Reduced reliance on frequent audits, resulting in cost savings for your organization
- Enhancement of risk management and control
- Compliance with audit requirements

HITRUST Compliance Consulting
- HITRUST Gap Analysis
- HITRUST Risk Assessment
- Security Awareness Training Program
- Documentation of HITRUST CSF Rules & Regulations
- HITRUST CSF Continuation Support

Coverage
- HITRUST CSF

PCI DSS Compliance Audit
- PCI DSS Gap Analysis
- PCI DSS Risk Assessment
- PCI DSS Penetration Testing
- PCI DSS ASV Scanning
- Security Awareness Training Program
- PCI Certification

Coverage
- PCI-DSS controls
- QSA led audits
- Support of SAQs
- Pre-audit readiness assessment

SOC 2 Advisory Services
- SOC2 Gap Analysis
- SOC2 Risk Assessment & Treatment
- SOC2 ISMS Implementation
- Security Awareness Training Program
- SOC2 Attestation
- SOC2 Continuation Support

Coverage
- SOC2 Type 1
- SOC2 Type 2
- AICPA Trust Services Principles

Reason
- Increased trust and transparency among internal and external stakeholders
- Compliance costs are reduced, and on-site audits are reduced.
- Assists in ensuring that controls are designed and implemented properly to mitigate risks.
- Compliance with auditing requirements

ISO Certification Advisory
- Gap Analysis of the Information Security Management System
- Risk Assessment of the Information Security Management System
- Services for ISMS Implementation
- Pre-Audit Services for Information Security Management Systems
- Training for ISO 27001 Certification
- Coordination and Assistance with ISO 27001 Third-Party Certification

Coverage
- ISO/IEC 27001:2013 controls

✓ Architecture Review and Assessment-

Security Automation and DevSecOps Consultation
- Secure Code Review
- Software Composition Analysis
- Change Management
- Configuration Management
- Threat Investigation
- Vulnerability Assessment
- Compliance Monitoring
- DevSecOps Security Training
- CI/CD Planning & Implementation

Coverage
- Static Application Security Testing (SAST)
- Dynamic Application Security Testing (DAST)
- Dynamic Application Security Testing (IAST)
- Software Composition Analysis (SCA)
- Penetration Testing
- Cloud Security Assessment
- Secure Development Training

Security Architecture and Configuration Reviews
- Cloud Architecture & Configuration Review
- Infrastructure Architecture & Configuration Review

Coverage
- Conduct a review of the most recent Threat Risk Analysis study.
- Analysis of the existing information technology network, information flow according to business
- requirements, and information access points.
- An analysis of existing security measures and policies in different areas of security management.
- Analyze the current network security architecture, including the topology/configuration and security
- components/functionality.

Threat Modelling
- Threat Modelling

Coverage
- Threat Landscape Assessment
- Attack Surface Assessment
- Complete Threat Hunting

Reason
The most efficient method of achieving the following goals is through threat modelling.
- Early in the SDLC, even before any code is written, identify issues.
- Identify design defects that conventional testing methods and code reviews may miss, and maximize
- your testing budget by assisting you in focusing your testing and code review efforts.
- Consider novel attack strategies that you may have overlooked previously.

- Identify requirements process flaws and save money by resolving issues prior to publishing software or rewriting expensive code.

Managed Compliance Services

In-depth Security|Penetration Testing

- Services
  - Advisory
  - Assessment
    - Gap Assessment
    - Risk Assessment
    - Internal Audit
    - Compliance & Regulatory Audit
  - Testing
    - Penetration Testing
    - Vulnerability Assessment
    - Load & Performance Testing
    - Configuration Audit
    - Log Analysis
    - Source Code Review
    - Black, Grey & White box testing
    - Security Testing Frameworks
  - Trainings
- Solutions
  - Application Security
    - Web Application Security
    - Mobile Application Security
    - Web Services & API Security
    - Thick Client
    - ERP / SAP Security Testing
  - IT/IoT/OT Infra Security
    - Network & Security Devices
    - VoIP
    - Wireless
    - IoT/OT/SCADA/ICS
    - Endpoint Security
  - Cloud Security
  - Social Engineering
    - Red Teaming
    - Phishing
  - Digital Cyber Forensic
  - DevSecOps
    - Container Security
    - Kubernetes
  - Managed Services
    - Managed Security
    - vCISO
    - DPO
    - For SMB
    - For Startups
    - Cyber Risk Management
    - Vendor / Supply Chain Cyber Security
    - Security Posture Maturity Assessment
    - GRC Automation
- SecurityScorecard ®
- Compliance
  - Compliance Management System
    - ISMS - ISO 27001
    - BCMS - ISO 22301
    - PIMS - ISO 27701
    - ERM - ISO 31000
    - SCSMS - ISO 28001
    - AMS - ISO 55001
    - CoBIT
    - ITSM - ISO 20000 / ITIL
    - ISA / IEC 62443
  - Compliance Assessment Frameworks
    - GDPR
    - CMMC
    - PCI DSS
    - SOC (1,2&3)
    - HIPAA
    - HITRUST
    - C2M2
    - TISAX
    - NIST
    - ITGC
    - COSO
    - CIS
    - CSA & CCM
    - TSS
    - FAIR
    - ISF-SOGP
    - ITU-CIIP
    - IOTCA
    - sa
    - MITRE ATTACK
    - SAMA
  - Regulatory Compliance
    - RBI
    - IREDA
    - IT Act
    - Cert-In
    - SEBI
    - TRAI
    - UIDAI
    - CERC
    - MAS Compliance
    - HKMA
    - NESA
    - SAR
    - DLP
    - PPI
    - NHB
    - VSCC
    - ISNP
    - DoT
    - NCIIPC
    - CII/CCMP
- Industries
- Company
  - About Us
  - Associate with Us
  - Get a Quote
  - Partners & Client
  - Contact Us
- Resources
  - Blog
  - Datasheets
  - Case Studies
  - White Papers

PenTesting Checklist Tools

Reconnaissance |Information gathering
General
Google Search - https://www.google.com/
Shodan - https://www.shodan.io/
Maltego - https://www.maltego.com/
Recon-ng - https://github.com/lanmaster53/recon-ng
theHarvester - https://github.com/laramies/theHarvester
FOCA - https://github.com/ElevenPaths/FOCA
SpiderFoot - https://www.spiderfoot.net/
OSINT Framework - https://osintframework.com/
Metagoofil - https://github.com/laramies/metagoofil
Tinfoleak - https://github.com/vaguileradiaz/tinfoleak

Portscanning tools
Google Search - https://www.google.com/
Shodan - https://www.shodan.io/
Maltego - https://www.maltego.com/
Recon-ng - https://github.com/lanmaster53/recon-ng
theHarvester - https://github.com/laramies/theHarvester
FOCA - https://github.com/ElevenPaths/FOCA
SpiderFoot - https://www.spiderfoot.net/
OSINT Framework - https://osintframework.com/
Metagoofil - https://github.com/laramies/metagoofil
Tinfoleak - https://github.com/vaguileradiaz/tinfoleak
Nmap - https://nmap.org/
Naabu - https://github.com/projectdiscovery/naabu
Masscan - https://github.com/robertdavidgraham/masscan
Netcat - http://www.stearns.org/nc/
AngryIP - https://angryip.org/

Open Source Intelligence (OSINT)
Censys - https://censys.io/
Foca Pro - https://www.elevenpaths.com/es/soluciones/foca-pro/index.html
Datasploit - https://github.com/DataSploit/datasploit
Sherlock - https://github.com/sherlock-project/sherlock
PhoneInfoga - https://github.com/sundowndev/PhoneInfoga
Sn1per - https://github.com/1N3/Sn1per
Photon - https://github.com/s0md3v/Photon
Infoga - https://github.com/m4ll0k/Infoga
Gitrob - https://github.com/michenriksen/gitrob
Amass - https://github.com/OWASP/Amass

Checking if domain live
Ping - https://en.wikipedia.org/wiki/Ping_(networking_utility)
Curl - https://curl.se/ Wget - https://www.gnu.org/software/wget/
Telnet - https://en.wikipedia.org/wiki/Telnet
Netcat - https://en.wikipedia.org/wiki/Netcat
Traceroute - https://en.wikipedia.org/wiki/Traceroute
Nmap - https://nmap.org/
Hping - http://www.hping.org/

Subdomain flyover tools
Sublist3r - https://github.com/aboul3la/Sublist3r
Recon-ng - https://github.com/lanmaster53/recon-ng
Amass - https://github.com/OWASP/Amass
Subfinder - https://github.com/projectdiscovery/subfinder
Aquatone - https://github.com/michenriksen/aquatone
Knockpy - https://github.com/guelfoweb/knock
theHarvester - https://github.com/laramies/theHarvester
Subbrute - https://github.com/TheRook/subbrute
dnsrecon - https://github.com/darkoperator/dnsrecon
EyeWitness - https://github.com/FortyNorthSecurity/EyeWitness
Fierce - https://github.com/mschwager/fierce

Code review
ESLint - https://eslint.org/ JSHint - https://jshint.com/
Retire.js - https://github.com/RetireJS/retire.js
Brakeman - https://brakemanscanner.org/
SonarJS - https://www.sonarqube.org/features/security/
Snyk - https://snyk.io/
jsprime - https://github.com/dpnishant/jsprime
QL for Javascript - https://www.semanticscholar.org/paper/QL-for-JavaScript%3A-Principles-and-Applications-Ferrari-Palomba/b2a73e71b3c2dcebe83f16a75711c290b8032029
CodeQL - https://securitylab.github.com/tools/codeql/
Jscrambler - https://jscrambler.com/

Content discovery
DirBuster - https://sourceforge.net/projects/dirbuster/
DirSearch - https://github.com/maurosoria/dirsearch
Gobuster - https://github.com/OJ/gobuster
wfuzz - https://github.com/xmendez/wfuzz
Arjun - https://github.com/s0md3v/Arjun
ffuf - https://github.com/ffuf/ffuf
Dirb - https://sourceforge.net/projects/dirb/
Nikto - https://cirt.net/nikto2
Wfuzz - https://github.com/xmendez/wfuzz
Skipfish - https://github.com/spinkham/skipfish

Vulnerability scan
1. Web app vulnerability scans
Burp suite pro
OWASP ZAP
Nuclei
Nikto
Acunetix
Nessus
Netsparker
OpenVAS
Qualys Web Application Scanning
Rapid7 AppSpider
Security Compass
Trustwave App Scanner
Veracode Dynamic Analysis

2. API vulnerability scanners
OWASP ZAP - https://www.zaproxy.org/
Postman - https://www.postman.com/api-security/
Insomnia - https://insomnia.rest/ SoapUI - https://www.soapui.org/api-testing/
Burp Suite - https://portswigger.net/burp/api-scanning
Rest-Assured - https://github.com/rest-assured/rest-assured/wiki/Usage
Checkmarx - https://www.checkmarx.com/products/api-security-testing/
Fortify - https://www.microfocus.com/en-us/products/fortify-application-security-testing/overview/api-testing
Netsparker - https://www.netsparker.com/api-security/
SecureLayer7 - https://www.securelayer7.net/api-security-testing/

External tools
1. Out of band servers
RequestBin - https://requestbin.com/
ngrok - https://ngrok.com/
Hookbin - https://hookbin.com/
Beeceptor - https://beeceptor.com/
Pipedream - https://pipedream.com/
Webhook.site - https://webhook.site/

2. Fuzzers
AFL (American Fuzzy Lop) - http://lcamtuf.coredump.cx/afl/
Peach Fuzzer - https://peachfuzzer.com/
Spike - https://github.com/aramosf/spike
Sulley - https://github.com/OpenRCE/sulley
OWASP ZAP Fuzzer - https://www.zaproxy.org/docs/desktop/addons/fuzzer/
Wfuzz - https://github.com/xmendez/wfuzz
Fuzzbox - https://github.com/fuzzbox-ws/fuzzbox
Radamsa - https://github.com/aoh/radamsa
BooFuzz - https://github.com/jtpereyda/boofuzz
Powerfuzzer - https://github.com/carlosmiranda/powerfuzzer
PeachPy - https://github.com/peachpiecompiler/peachpy

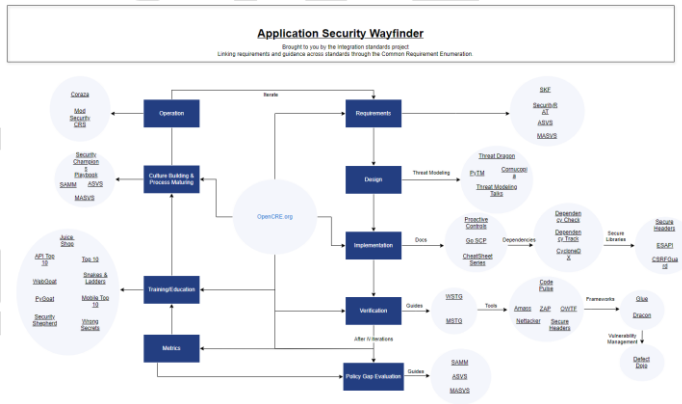
3. XSS scanners
Acunetix - https://www.acunetix.com/
AppScan - https://www.ibm.com/security/application-security/appscan
Arachni - https://www.arachni-scanner.com/
Burp Suite - https://portswigger.net/burp
Detectify - https://detectify.com/
Fortify - https://www.microfocus.com/en-us/products/application-security-testing/overview
IBM Security AppScan - https://www.ibm.com/security/application-security/appscan
Netsparker - https://www.netsparker.com/
Nikto - https://cirt.net/Nikto2
OWASP ZAP - https://www.zaproxy.org/
Qualys Web Application Scanning - https://www.qualys.com/apps/web-app-scanning/
Retina - https://www.beyondtrust.com/products/retina-network-security-scanner/
Rapid7 AppSpider - https://www.rapid7.com/products/appspider/
Skipfish - https://tools.kali.org/web-applications/skipfish
Vega - https://subgraph.com/vega/
Veracode Dynamic Analysis - https://www.veracode.com/products/dynamic-analysis

4. Blind XSS
XSS Hunter - https://xsshunter.com/
Blind XSS Injector - https://github.com/cujanovic/Blind-XSS-Injector
Sleepy Puppy - https://github.com/Netflix/sleepy-puppy
XSStrike - https://github.com/s0md3v/XSStrike
XSStrumer - https://github.com/Manas-Harsh/XSStrumer
Wfuzz - https://github.com/xmendez/wfuzz
BruteXSS - https://github.com/shawarkhanethicalhacker/BruteXSS
Dalfox - https://github.com/hahwul/dalfox

Fiddler - https://www.telerik.com/fiddler
Burp Suite - https://portswigger.net/burp

5. SQL scanners
SQLMap - http://sqlmap.org/
Havij - http://www.itsecteam.com/en/projects/project1.htm
DorkNet - https://github.com/NullArray/DorkNet
NoSQLMap - https://github.com/codingo/NoSQLMap
jSQL Injection - https://github.com/ron190/jsql-injection
Blind SQL Injector - https://github.com/UltimateHackers/sqlmate
SQLMate - https://github.com/UltimateHackers/sqlmate
SQLNinja - https://github.com/sqlninja/sqlninja
BBQSQL - https://github.com/Neohapsis/bbqsql
SQLSentinel - https://github.com/xxlegendriderx/SQLSentinel

OWASP Projects, the SDLC, and the Security Wayfinder



Flagship Projects

- OWASP Amass

  An open source framework that helps information security professionals perform network mapping of attack surfaces and external asset discovery using open source intelligence gathering and reconnaissance techniques!

- OWASP Application Security Verification Standard

  The OWASP Application Security Verification Standard (ASVS) Project is a framework of security requirements that focus on defining the security controls required when designing, developing and testing modern web applications and web services.

- OWASP Cheat Sheet Series

  The OWASP Cheat Sheet Series project provides a set of concise good practice guides for application developers and defenders to follow.

- OWASP CycloneDX

  OWASP CycloneDX is a full-stack Bill of Materials (BOM) standard that provides advanced supply chain capabilities for cyber risk reduction.

- OWASP Defectdojo

  The leading open source application vulnerability management tool built for DevOps and continuous security integration.

- OWASP Dependency-Check

  Dependency-Check is a Software Composition Analysis (SCA) tool suite that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.

- OWASP Dependency-Track

  Intelligent Component Analysis platform that allows organizations to identify and reduce risk in the software supply chain.

- OWASP Juice Shop

  Probably the most modern and sophisticated insecure web application for security trainings, awareness demos and CTFs. Also great voluntary guinea pig for your security tools and DevSecOps pipelines!

- OWASP Mobile Application Security

  The OWASP Mobile Application Security (MAS) project consists of a series of documents that establish a security standard for mobile apps and a comprehensive testing guide that covers the processes, techniques, and tools used during a mobile application security

assessment, as well as an exhaustive set of test cases that enables testers to deliver consistent and complete results.

- OWASP ModSecurity Core Rule Set

  The OWASP ModSecurity Core Rule Set (CRS) is a set of generic attack detection rules for use with ModSecurity or compatible web application firewalls. The CRS aims to protect web applications from a wide range of attacks, including the OWASP Top Ten, with a minimum of false alerts.
- OWASP OWTF

  Offensive Web Testing Framework (OWTF), is an OWASP+PTES focused try to unite great tools and make pen testing more efficient, written mostly in Python.
- OWASP SAMM

  A Software Assurance Maturity Model (SAMM) that provides an effective and measurable way for all types of organizations to analyse and improve their software security posture.
- OWASP Security Shepherd

  OWASP Security Shepherd is a web and mobile application security training platform. Security Shepherd has been designed to foster and improve security awareness among a varied skill-set demographic. The aim of this project is to take AppSec novices or experienced engineers and sharpen their penetration testing skillset to security expert status.
- OWASP Top Ten

  The OWASP Top 10 is the reference standard for the most critical web application security risks. Adopting the OWASP Top 10 is perhaps the most effective first step towards changing your software development culture focused on producing secure code.
- OWASP Web Security Testing Guide

  The Web Security Testing Guide (WSTG) Project produces the premier cybersecurity testing resource for web application developers and security professionals.

Production Projects

- OWASP API Security Project
- OWASP Bug Logging Tool

  OWASP BLT is a tool enabling internet users to report all kinds of issues they encounter, thereby improving internet security, with a unique feature of rewarding users for bug reporting and allowing companies to launch their own bug hunting programs, promoting responsible disclosure and fostering a safer online environment.
- OWASP Coraza Web Application Firewall

  OWASP Coraza is a golang enterprise-grade WAF framework compatible with Modsecurity and OWASP Core Ruleset.
- OWASP CSRFGuard

  OWASP CSRFGuard is a library that implements a variant of the synchronizer token pattern to mitigate the risk of Cross-Site Request Forgery (CSRF) attacks.
- OWASP ModSecurity

  ModSecurity is the standard open-source web application firewall (WAF) engine.
- OWASP WrongSecrets

  Examples with how to not use secrets

A-Z Computer security exploits

A

- Account pre-hijacking
- Armitage (computing)

B

- Badlock
- BadUSB
- Blended threat
- Blind return oriented programming
- BlueKeep
- Operation: Bot Roast
- Buffer over-read
- Buffer overflow
- Buffer overflow protection

C

- Cable Haunt
- Cache poisoning
- Clear channel assessment attack
- Cloudbleed
- Common Vulnerabilities and Exposures
- Copy attack
- Covert channel
- Cross-application scripting
- Cyber Insider Threat
- Cybersecurity Capacity Maturity Model for Nations

D

- Dangling pointer
- Default password
- Dirty COW
- DirtyTooth
- DNS leak
- DNS spoofing
- DoublePulsar
- Downfall (security vulnerability)
- Drive-by download
- DROWN attack
- Dynamic linker

E

- EFAIL
- *EgotisticalGiraffe*
- EternalBlue
- Evasion (network security)
- Evil maid attack
- Exploit kit

F

- Uncontrolled format string

- FragAttacks

G

- GoFetch

H

- Hardware security bug
- Heap feng shui
- Heap overflow
- Heap spraying
- Heartbleed
- Hertzbleed
- HTTP parameter pollution
- Hyperjacking

I

- Idle scan
- Improper input validation
- In-session phishing
- Integer overflow
- Intrusion detection system evasion techniques
- Ivanti Pulse Connect Secure data breach

J

- JIT spraying

K

- KARMA attack
- Kr00k
- KRACK

L

- Laptop theft
- Log4Shell
- Login spoofing
- LogoFAIL
- Lorcon

M

- Meltdown (security vulnerability)
- Memory corruption
- Memory safety
- Mimikatz
- Misfortune Cookie (computers)
- Mixed threat attack

N

- Network Investigative Technique
- NOP slide
- Null character
- Null session

O

- Off-by-one error
- Open Bug Bounty

P

- Pass the hash
- Payload (computing)
- Pharming
- POODLE
- Phillip Porras
- Port scanner
- Predictable serial number attack
- PrintNightmare
- Project 25
- *Prompt injection*

R

- Race condition
- Racetrack problem
- Ramsay Malware
- Reflection attack
- Register spring
- Relay attack
- Reptar (vulnerability)
- Return-oriented programming
- Return-to-libc attack
- Ripple20
- Row hammer
- Ryuk (ransomware)

S

- Server-side request forgery
- Shatter attack
- Shellshock (software bug)
- SIGRed
- Sigreturn-oriented programming
- SigSpoof
- Silver Sparrow (malware)
- SMBRelay
- SMS spoofing
- Smudge attack
- Spectre (security vulnerability)
- Stack buffer overflow
- Stagefright (bug)
- Structural vulnerability (computing)
- Swatting
- Symlink race
- System Reconfiguration Attacks

T

- TCP reset attack
- Terrapin attack
- Threat (computer)
- Threat model
- Time-of-check to time-of-use
- Transient execution CPU vulnerability
- TRESOR

V

- Vault 7
- VENOM
- Virtual machine escape
- Computer virus
- FluBot
- Source code virus
- Virus hoax
- Voice phishing
- Vulnerability database

W

- WannaCry ransomware attack
- WAP billing
- Warchalking
- Wardialing
- Wardriving

- Warshipping
- Webattacker
- Weird machine
- Windows Metafile vulnerability
- WooYun

X

- XZ Utils backdoor

Z

- *Zero-click attack*
- Zeroday Emergency Response Team
- Zerodium

Special Edition

C

1. Cryptographic attacks (5 C, 117 P)
1.1. Attacks on public-key cryptosystems (1 C, 4 P)
1.1.1. RSA Factoring Challenge (3 P)
1.2. Chosen-plaintext attacks (3 P)
1.3. Cryptanalytic software (4 C, 6 P)
1.3.1. Digital rights management circumvention software (1 C, 8 P)
1.3.1.1. DVD rippers (13 P)
1.3.2. DVD rippers (13 P)
1.3.3. Integer factorization algorithms (26 P)
1.3.4. Password cracking software (15 P)
1.4. Password cracking software (15 P)
1.5. Side-channel attacks (1 C, 27 P)
1.5.1. Speculative execution security vulnerabilities (14 P)

D

2. Denial-of-service attacks (1 C, 78 P)
2.1. Algorithmic complexity attacks (4 P)

I

3. Injection exploits (22 P)

M

4. Malware (5 C, 40 P)
4.1. Malware by type (14 C)
4.1.1. Types of malware (44 P)
4.1.2. Adware (1 C, 40 P)
4.1.2.1. Windows adware (5 P)
4.1.3. Botnets (1 C, 67 P)
4.1.3.1. Computer security companies specializing in botnets (6 P)
4.1.4. Computer surveillance (3 C, 30 P)
4.1.4.1. Edward Snowden (2 C, 25 P)
4.1.4.1.1. Cultural depictions of Edward Snowden (8 P)
4.1.4.1.2. Intelligence agency programmes revealed by Edward Snowden (22 P)
4.1.4.2. Spyware (5 C, 65 P)
4.1.4.2.1. Accountability software (2 P)
4.1.4.2.2. Spyware companies (9 P)
4.1.4.2.3. Spyware removal (1 C, 22 P)
4.1.4.2.3.1. Screenshots of spyware removal (7 F)
4.1.4.2.4. Spyware used by governments (9 P)
4.1.4.2.5. Stalkerware (8 P)
4.1.4.3. Spyware companies (9 P)
4.1.5. Computer viruses (8 C, 37 P)
4.1.5.1. Amiga viruses (3 P)
4.1.5.2. Antivirus software (5 C, 87 P)
4.1.5.2.1. Antivirus software for DOS (6 P)
4.1.5.2.2. Antivirus software for Linux (1 C, 5 P)
4.1.5.2.2.1. Antivirus software that uses Qt (1 P)
4.1.5.2.3. Free antivirus software (3 P)
4.1.5.2.4. Norton (software) (21 P)
4.1.5.2.5. Screenshots of antivirus software (12 F)
4.1.5.3. DOS viruses (3 C)
4.1.5.3.1. Antivirus software for DOS (6 P)
4.1.5.3.2. Boot viruses (11 P)
4.1.5.3.3. DOS file viruses (29 P)
4.1.5.4. Fictional computer viruses (1 C, 12 P)
4.1.5.4.1. Virus hoaxes (9 P)
4.1.5.5. Linux viruses (2 P)
4.1.5.6. Classic Mac OS viruses (8 P)
4.1.5.7. Macro viruses (1 P)
4.1.5.8. Windows viruses (1 C, 1 P)
4.1.5.8.1. Windows file viruses (8 P)
4.1.6. Computer worms (2 C, 51 P)
4.1.6.1. Email worms (27 P)
4.1.6.2. Exploit-based worms (21 P)
4.1.7. IoT malware (9 P)
4.1.8. Malware toolkits (22 P)
4.1.9. Ransomware (41 P)
4.1.10. Rogue software (1 C, 36 P)
4.1.10.1. Scareware (20 P)
4.1.11. Rootkits (2 C, 31 P)
4.1.11.1. Rootkit detection software (4 P)
4.1.11.2. Windows rootkit techniques (2 P)
4.1.12. Scareware (20 P)
4.1.13. Spyware (5 C, 65 P)
4.1.13.1. Accountability software (2 P)
4.1.13.2. Spyware companies (9 P)
4.1.13.3. Spyware removal (1 C, 22 P)
4.1.13.3.1. Screenshots of spyware removal (7 F)
4.1.13.4. Spyware used by governments (9 P)
4.1.13.5. Stalkerware (8 P)
4.1.14. Trojan horses (3 C, 56 P)
4.1.14.1. Common trojan horse payloads (4 P)
4.1.14.2. Software that bundles malware (5 P)
4.1.14.3. Windows trojans (61 P)
4.2. Fiction about malware (6 C, 80 P)
4.2.1. Digimon (5 C, 10 P)
4.2.1.1. Digimon anime and manga (1 C, 11 P)
4.2.1.1.1. Digimon films (10 P)
4.2.1.2. Digimon character redirects to lists (373 P)
4.2.1.3. Digimon media files (1 C, 57 F)
4.2.1.3.1. Digimon anime DVD covers (6 F)
4.2.1.4. Digimon video games (1 C, 17 P)
4.2.1.4.1. Digimon spin-off games (10 P)
4.2.1.5. Digimon lists (2 C, 2 P)
4.2.1.5.1. Lists of Digimon characters (4 P)
4.2.1.5.2. Digimon episode lists (13 P)
4.2.2. Fictional computer viruses (1 C, 12 P)
4.2.3. Ghost in the Shell (6 C, 3 P)
4.2.4. The Matrix (franchise) (2 C, 14 P)
4.2.5. Television episodes about malware (7 P)
4.2.6. Tom Clancy's Net Force (6 P
4.3. Malware by platform (4 C, 3 P)
4.3.1. Linux malware (1 C, 13 P)
4.3.1.1. Linux viruses (2 P)
4.3.2. MacOS malware (1 C, 11 P)
4.3.2.1. Classic Mac OS viruses (8 P)
4.3.3. Mobile malware (2 C, 8 P)