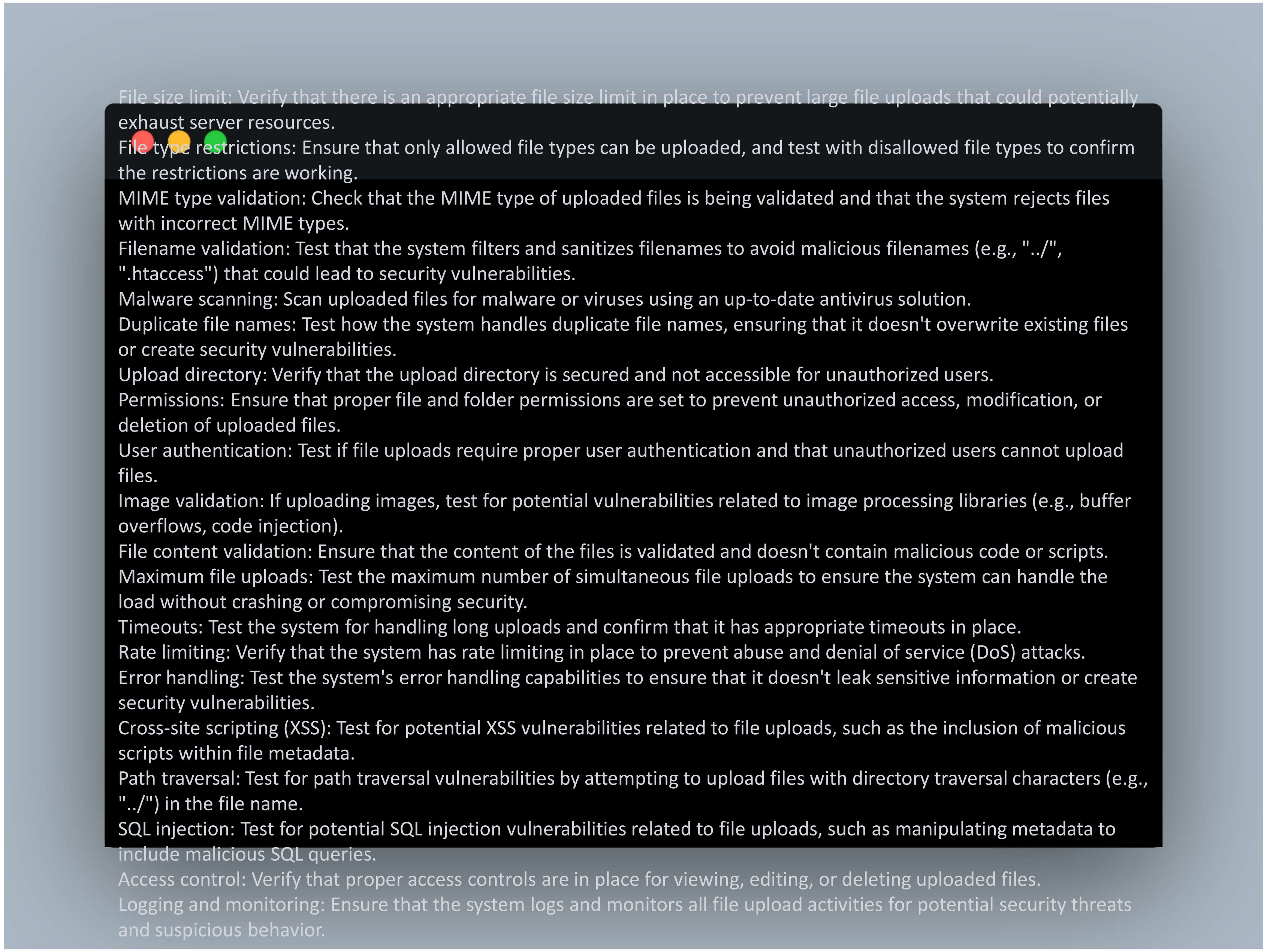

File size limit: Verify that there is an appropriate file size limit in place to prevent large file uploads that could potentially exhaust server resources.
File type restrictions: Ensure that only allowed file types can be uploaded, and test with disallowed file types to confirm the restrictions are working.
MIME type validation: Check that the MIME type of uploaded files is being validated and that the system rejects files with incorrect MIME types.
Filename validation: Test that the system filters and sanitizes filenames to avoid malicious filenames (e.g., "../", ".htaccess") that could lead to security vulnerabilities.
Malware scanning: Scan uploaded files for malware or viruses using an up-to-date antivirus solution.
Duplicate file names: Test how the system handles duplicate file names, ensuring that it doesn't overwrite existing files or create security vulnerabilities.
Upload directory: Verify that the upload directory is secured and not accessible for unauthorized users.
Permissions: Ensure that proper file and folder permissions are set to prevent unauthorized access, modification, or deletion of uploaded files.
User authentication: Test if file uploads require proper user authentication and that unauthorized users cannot upload files.
Image validation: If uploading images, test for potential vulnerabilities related to image processing libraries (e.g., buffer overflows, code injection).
File content validation: Ensure that the content of the files is validated and doesn't contain malicious code or scripts.
Maximum file uploads: Test the maximum number of simultaneous file uploads to ensure the system can handle the load without crashing or compromising security.
Timeouts: Test the system for handling long uploads and confirm that it has appropriate timeouts in place.
Rate limiting: Verify that the system has rate limiting in place to prevent abuse and denial of service (DoS) attacks.
Error handling: Test the system's error handling capabilities to ensure that it doesn't leak sensitive information or create security vulnerabilities.
Cross-site scripting (XSS): Test for potential XSS vulnerabilities related to file uploads, such as the inclusion of malicious scripts within file metadata.
Path traversal: Test for path traversal vulnerabilities by attempting to upload files with directory traversal characters (e.g., "../") in the file name.
SQL injection: Test for potential SQL injection vulnerabilities related to file uploads, such as manipulating metadata to include malicious SQL queries.
Access control: Verify that proper access controls are in place for viewing, editing, or deleting uploaded files.
Logging and monitoring: Ensure that the system logs and monitors all file upload activities for potential security threats and suspicious behavior.

INSECURE DESERIALIZATION CHECKLIST

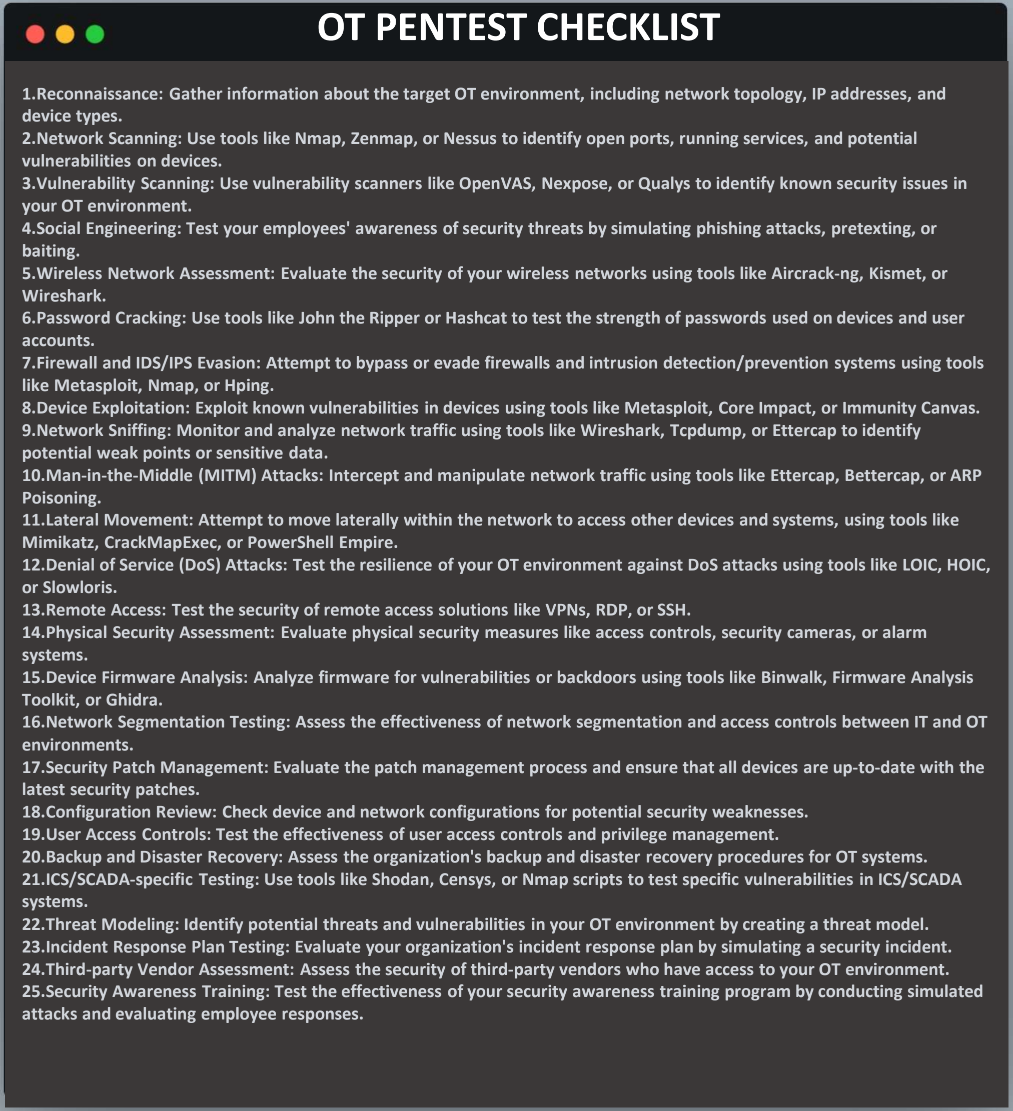
- Input validation: Validate and sanitize all input data, including any data that is deserialized.
- Data type checks: Verify that the data being deserialized matches the expected data type.
- Input filtering: Filter any input data that is not expected or not needed for the deserialization process.
- Object graph integrity: Verify that the object graph is not tampered with during the deserialization process.
- Object instantiation: Ensure that only authorized classes are being instantiated during deserialization.
- Input length restrictions: Limit the size of the input data being deserialized to prevent overflow attacks.
- Type safety: Use strongly typed languages to avoid type confusion attacks.
- Content type checks: Verify that the content type of the serialized data is as expected.
- Whitelisting: Use a whitelist to allow only known and expected serialized classes.
- Signature verification: Verify the digital signature of the serialized data to ensure it has not been tampered with.
- Strict deserialization: Use strict deserialization to prevent deserialization of untrusted data.
- Sandbox environments: Use sandboxed environments to test deserialization vulnerabilities in a safe environment.
- Behavior-based analysis: Analyze the behavior of the deserialized data to detect any unusual or unexpected behavior.
- Testing for code injection: Test for code injection attacks that exploit insecure deserialization.
- Input parameter testing: Test for input parameter attacks that exploit insecure deserialization.
- Error handling testing: Test the error handling mechanisms to ensure they do not leak sensitive information.
- Integration testing: Test the integration of the deserialization process with other components.
- Monitoring and logging: Monitor and log deserialization activities to detect and respond to potential attacks.
- Patching and updates: Keep your systems up to date with the latest security patches and updates.
- Threat modeling: Conduct threat modeling exercises to identify and mitigate insecure deserialization risks in your system.
- Application security testing: Conduct application security testing to identify and remediate insecure deserialization vulnerabilities.
- Reverse engineering: Use reverse engineering techniques to understand how the deserialization process works.
- Code review: Conduct code reviews to identify insecure deserialization vulnerabilities in the code.
- Dependency analysis: Analyze the dependencies of the deserialization library and ensure they are up to date.
- Security training: Train developers and other stakeholders on secure coding practices and the risks of insecure deserialization.

https://www.linkedin.com/in/joas-antonio-dos-santos

FIREWALL TESTING CHECKLIST

1.Port scanning: Tool: Nmap (https://nmap.org/)
2.OS fingerprinting: Tool: Xprobe2 (http://xprobe.sourceforge.net/)
3.Firewall rule testing: Tool: Firewalk (https://github.com/defunkt/firewalk)
4.Packet fragmentation evasion: Tool: Fragroute (https://github.com/plitex/fragroute)
5.IP spoofing: Tool: Hping3 (https://github.com/antirez/hping)
6.Protocol-specific evasion: Tool: Metasploit Framework (https://www.metasploit.com/)
7.ICMP tunneling: Tool: ICMPTX (http://thomer.com/icmptx/)
8.DNS tunneling: Tool: Dns2tcp (https://github.com/alex-sector/dns2tcp)
9.HTTP tunneling: Tool: HTTPTunnel (https://github.com/larsbrinkhoff/httptunnel)
10.IPv6 tunneling: Tool: Teredo (https://tools.ietf.org/html/rfc4380)
11.ARP spoofing: Tool: Ettercap (https://www.ettercap-project.org/)
12.SSL/TLS interception: Tool: SSLstrip (https://github.com/moxie0/sslstrip)
13.SSL/TLS decryption: Tool: Wireshark (https://www.wireshark.org/)
14.SSH tunneling: Tool: OpenSSH (https://www.openssh.com/)
15.Proxy server evasion: Tool: Proxychains (https://github.com/rofl0r/proxychains-ng)
16.TOR network evasion: Tool: Tor Browser (https://www.torproject.org/)
17.Web application firewall (WAF) testing: Tool: Wafw00f (https://github.com/EnableSecurity/wafw00f)
18.Session hijacking: Tool: Cookie Cadger (https://github.com/cookiecadger/CookieCadger)
19.Man-in-the-middle attack: Tool: Bettercap (https://www.bettercap.org/)
20.VPN detection: Tool: Iodine (https://github.com/yarrick/iodine)
21.Firewall evasion using encrypted payloads: Tool: Veil-Evasion (https://github.com/Veil-Framework/Veil)
22.Application-level evasion using SQL injection: Tool: SQLMap (https://sqlmap.org/)
23.Application-level evasion using Cross-Site Scripting (XSS): Tool: XSSer (https://github.com/epsylon/xsser)
24.File type and extension evasion: Tool: FuzzDB (https://github.com/fuzzdb-project/fuzzdb)
25.Web service scanning and evasion: Tool: Nikto (https://github.com/sullo/nikto)

https://www.linkedin.com/in/joas-antonio-dos-santos



OT PENTEST CHECKLIST

1.Reconnaissance: Gather information about the target OT environment, including network topology, IP addresses, and device types.
2.Network Scanning: Use tools like Nmap, Zenmap, or Nessus to identify open ports, running services, and potential vulnerabilities on devices.
3.Vulnerability Scanning: Use vulnerability scanners like OpenVAS, Nexpose, or Qualys to identify known security issues in your OT environment.
4.Social Engineering: Test your employees' awareness of security threats by simulating phishing attacks, pretexting, or baiting.
5.Wireless Network Assessment: Evaluate the security of your wireless networks using tools like Aircrack-ng, Kismet, or Wireshark.
6.Password Cracking: Use tools like John the Ripper or Hashcat to test the strength of passwords used on devices and user accounts.
7.Firewall and IDS/IPS Evasion: Attempt to bypass or evade firewalls and intrusion detection/prevention systems using tools like Metasploit, Nmap, or Hping.
8.Device Exploitation: Exploit known vulnerabilities in devices using tools like Metasploit, Core Impact, or Immunity Canvas.
9.Network Sniffing: Monitor and analyze network traffic using tools like Wireshark, Tcpdump, or Ettercap to identify potential weak points or sensitive data.
10.Man-in-the-Middle (MITM) Attacks: Intercept and manipulate network traffic using tools like Ettercap, Bettercap, or ARP Poisoning.
11.Lateral Movement: Attempt to move laterally within the network to access other devices and systems, using tools like Mimikatz, CrackMapExec, or PowerShell Empire.
12.Denial of Service (DoS) Attacks: Test the resilience of your OT environment against DoS attacks using tools like LOIC, HOIC, or Slowloris.
13.Remote Access: Test the security of remote access solutions like VPNs, RDP, or SSH.
14.Physical Security Assessment: Evaluate physical security measures like access controls, security cameras, or alarm systems.
15.Device Firmware Analysis: Analyze firmware for vulnerabilities or backdoors using tools like Binwalk, Firmware Analysis Toolkit, or Ghidra.
16.Network Segmentation Testing: Assess the effectiveness of network segmentation and access controls between IT and OT environments.
17.Security Patch Management: Evaluate the patch management process and ensure that all devices are up-to-date with the latest security patches.
18.Configuration Review: Check device and network configurations for potential security weaknesses.
19.User Access Controls: Test the effectiveness of user access controls and privilege management.
20.Backup and Disaster Recovery: Assess the organization's backup and disaster recovery procedures for OT systems.
21.ICS/SCADA-specific Testing: Use tools like Shodan, Censys, or Nmap scripts to test specific vulnerabilities in ICS/SCADA systems.
22.Threat Modeling: Identify potential threats and vulnerabilities in your OT environment by creating a threat model.
23.Incident Response Plan Testing: Evaluate your organization's incident response plan by simulating a security incident.
24.Third-party Vendor Assessment: Assess the security of third-party vendors who have access to your OT environment.
25.Security Awareness Training: Test the effectiveness of your security awareness training program by conducting simulated attacks and evaluating employee responses.

https://www.linkedin.com/in/joas-antonio-dos-santos

SCADA SHODAN DORK

1.SCADA system product: Empresaxyz SCADA
2.SCADA system by port: port:502 Empresaxyz
3.SCADA system by protocol: Empresaxyz Modbus
4.SCADA system by specific keyword: Empresaxyz HMI
5.SCADA system with a specific OS: Empresaxyz "Windows XP"
6.SCADA system with a specific manufacturer: Empresaxyz "Siemens PLC"
7.SCADA system with a specific software: Empresaxyz "Wonderware"
8.SCADA system with a specific device type: Empresaxyz "RTU"
9.SCADA system with a specific service: Empresaxyz "DNP3"
10.SCADA system with a specific city: Empresaxyz city:"New York"
11.SCADA system with a specific country: Empresaxyz country:"US"
12.SCADA system using a specific product version: Empresaxyz "FactoryTalk View SE"
13.SCADA system with a specific IP range: net:192.168.1.0/24 Empresaxyz
14.SCADA system with a specific hostname: hostname:scada.empresaxyz.com
15.SCADA system with default credentials: Empresaxyz "default password"
16.SCADA system with known vulnerabilities: Empresaxyz vuln:CVE-2020-12345
17.SCADA system with a specific organization: org:"Empresaxyz"
18.SCADA system with a specific ICS protocol: Empresaxyz "IEC 61850"
19.SCADA system with a specific web server: Empresaxyz "Apache"
20.SCADA system with a specific hardware: Empresaxyz "Schneider Electric"
21.SCADA system with exposed VNC: Empresaxyz port:5900
22.SCADA system with exposed RDP: Empresaxyz port:3389
23.SCADA system with exposed OPC UA: Empresaxyz port:4840
24.SCADA system with exposed Profinet: Empresaxyz port:34962
25.SCADA system with exposed EtherNet/IP: Empresaxyz port:44818

https://www.linkedin.com/in/joas-antonio-dos-santos

ADVERSARY EMULATION CAMPAIGN HINT

1.Understand Your Objectives: Your primary aim is to mimic the tactics, techniques, and procedures (TTPs) of real-life adversaries to test and strengthen your organization's defenses. Make sure your objectives align with this.
2. Scope Appropriately: Define what is within the scope of your operation and stick to it. This includes systems, networks, and physical spaces.
3.Leverage Threat Intelligence: Use up-to-date threat intelligence to make your emulation as realistic as possible. Choose an adversary whose TTPs you will emulate.
4.Follow the ATT&CK Framework: The MITRE ATT&CK framework is a great guide for emulating adversary behavior. It provides comprehensive information about different TTPs.
5.Multi-Vector Attacks: Real adversaries won't limit themselves to just one vector. Consider including multiple attack vectors in your emulation.
6.Emphasize Stealth: Adversaries will typically try to avoid detection. Your red team should emulate this by using stealthy techniques and avoiding unnecessary noise.
7.Use Tools Wisely: Use a blend of off-the-shelf tools, custom software, and manual techniques. Remember, it's about emulating the adversary, not the tools they use.
8.Practice Safe Operations: Make sure you're not causing actual harm to your organization. Always have a rollback plan in case something goes wrong.
9.Include Social Engineering: Many adversaries use social engineering techniques. Including these in your emulation can make it more realistic and test your human defenses.
10.Regularly Update Skills: The cybersecurity landscape is continually evolving, and so are adversaries. Regular training and education, like the courses offered by SANS, Zero Point, SpecterOps and Others, can help you keep up.
11.Test Incident Response: Your emulation should not only test your defenses but also your response capabilities. How quickly and effectively can your organization respond to a breach?
12.Use Deception: Plant false flags or deceptive information to emulate sophisticated adversaries and test your blue team's analytical capabilities.
13.Real-Time Adjustments: Monitor the operation and make real-time adjustments as necessary. A real adversary would change their tactics if they were not working, and so should you.
14.Post-Operation Analysis: After the operation, conduct a thorough analysis. What worked, what didn't, and why?
15.Share Knowledge: Lessons learned should be shared across your organization to improve overall security. Use the red team operation as a learning tool, not just a test.

Tools: Atomic Red Team, Cobalt Strike, Caldera, RTA, Infect Monkey, Covenant or SliverC2

https://www.linkedin.com/in/joas-antonio-dos-santos