1. P&J Benedict Security methodology for Penetration Test

P&J Benedict Security uses a specific methodology to assess the security level of organizations in a uniform and consistent way. Our methodology is a combination of our know-how obtained by executing security tests for various types of organizations and is based on open standards including the Open Web Application Security Project, the Open Source Security Testing Methodology Manual and the ISO 17799 standard.

Our methodology

We follow a step-by-step approach to security testing:

1. **The first step is threat modelling:**

Threat modelling is a lightweight approach to assess the risk and exposure of an organization. We need to identify the assets that are important for your business and how these assets are currently protected. Assets can be: customers stored in a database, a webmail application, an Extranet application, your wireless network,…

2. **The second step is reconnaissance:**

In a passive way we attempt to find out more information about the assets identified in Step 1. This can include the screening of public databases, Google, newsgroups, social engineering,… This results in possible targets that provide entry points to the assets from phase 1.

3. **The third step is information gathering:**

In an active way we access the targets from step 2 and learn more about how the assets are protected. We detect all accessible ports, services and applications running on the targets.

4. **The fourth step is vulnerability detection:**

We identify all possible vulnerabilities in the targets like SQL Injection, Cross-site-scripting, buffer overflow, out-of-date services, default passwords, lack of authorization controls and many more. This can be accomplished by testing the infrastructure and applications and can include source-code review.

5. **The fifth step is the attack phase:**

We exploit the vulnerabilities from Step 4 without invading the integrity of the targets. The exploitation step allows to demonstrate the impact of discovered vulnerabilities.

6. **The final step is the reporting and presentation phase:**

Each vulnerability is documented in a vulnerability report with the corresponding impact. This report includes the steps required to reproduce the exploit. We propose countermeasures to solve or to mitigate the vulnerability in a cost-efficient way. A presentation to the project sponsor is foreseen to finalize the security test.

Our methodology consists of different attacks on the infrastructure and application layer, targeting development flaws within the application and configuration errors. These attacks are described in detail below.

- **Authentication/Authorization**

Risks in this area include access to the system by an unauthorized user, theft of usernames or passwords and password cracking/dictionary attacks, and ability to bypass authentication or authentication logging

1. **Password Cracking:** Password cracking is the process of validating password strength through the use of automated password recovery tools that expose either the application of weak cryptographic algorithms, incorrect implementation of cryptographic algorithms, or weak passwords due to human factors.
    - Obtain the password file from the system that stores usernames and passwords
    - Run an automated dictionary attack on the password file
    - Run a brute force attack on the password file as time and processing cycles allow
    - Use obtained passwords or their variations to access additional systems or applications
    - Run automated password crackers on encrypted files that are encountered (such as PDFs or Word documents) in an attempt to gather more intelligence and highlight the need for stronger document or file system encryption.
    - Verify password aging.
    - Launch brute force/dictionary attacks against systems like web applications, Telnet, FTP, … requiring authentication

2. **Authentication**

    a) Find possible brute force password guessing access points in the applications.
    b) Find a valid login credentials with password grinding, if possible.
    c) Bypass authentication system with spoofed tokens.
    d) Bypass authentication system with replay authentication information.
    e) Determine the application logic to maintain the authentication sessions - number of (consecutive) failure logins allowed, login timeout, etc.
    f) Determine the limitations of access control in the applications - access permissions, login session duration, idle duration.
    g) Session Management
    h) Determine the session management information - number of concurrent sessions, IP-based authentication, role-based authentication, identity-based authentication, cookie usage, session ID in URL encoding string, session ID in hidden HTML field variables, etc.
    i) Guess the session ID sequence and format
    j) Determine the session ID is maintained with IP address information; check if the same session information can be retried and reused in another machine.
    k) Gather excessive information with direct URL, direct instruction, action sequence jumping and/or pages skipping.
    l) Gather sensitive information with Man-In-the-Middle attacks.
    m) Inject excess/bogus information with Session-Hijacking techniques.
    n) Replay gathered information to fool the applications.

- **Privilege Escalation**

Risks in this area include users gaining access to roles, functions or information that are not intended for them by bypassing the privilege or permission scheme within the application

- **Information Disclosure**

Risks in this area include users gaining access to information that does not belong to them including company data and other user's personal data, credit card numbers or account balances

1. Information Leakage

a) Find useful information in hidden field variables of the HTML forms and comments in the HTML documents.
b) Examine the information contained in the application banners, usage instructions, welcome messages, farewell messages, application help messages, debug/error messages, etc.

2. Output Manipulation

- Retrieve valuable information stored in the cookies
- Retrieve valuable information from the client application cache.
- Retrieve valuable information stored in the serialized objects.
- Retrieve valuable information stored in the temporary files and objects.

- **Tampering**

Risks in this area include a user deleting another user's data, modifying logs or financial information and installing rootkits or backdoors

Input Manipulation

- Find the limitations of the defined variables and protocol payload - data length, data type, construct format, etc.
- Use exceptionally long character-strings to find buffer overflows vulnerability in the applications.
- Concatenate commands in the input strings of the applications.
- Inject SQL language in the input strings of database-tired web applications.
- Examine "Cross-Site Scripting" in the web applications of the system.
- Examine unauthorized directory/file access with path/directory traversal in the input strings of the applications.
- Use specific URL-encoded strings and/or Unicode-encoded strings to bypass input validation mechanisms of the applications.
- Execute remote commands through "Server Side Include".
- Manipulate the session/persistent cookies to fool or modify the logic in the server-side web applications.
- Manipulate the (hidden) field variable in the HTML forms to fool or modify the logic in the server-side web applications.
- Manipulate the "Referrer", "Host", etc. HTTP Protocol variables to fool or modify the logic in the server side web applications.
- Use illogical/illegal input to test the application error-handling routines and to find useful debug/error messages from the applications.

All attacks and tests are executed manually. P&J Benedict Security only uses automated tools for specific functions like brute-forcing ports, passwords, identifiers,…

- **Tools**

The following tools can be used during the intrusive test:

- Commercial:

    - Suru: web proxy and fuzzing tool from Sensepost
    - BiDiBlah: penetration testing tool from Sensepost
    - Burp Suite: web security testing tool from Portswigger
    - QualysGuard Consultancy Edition

- Non-commercial:
    - Nessus: vulnerability scanner from Tenable Security
    - Nmap: open-source port scanner
    - Wikto: brute force enumeration tool from Sensepost
    - Crowbar: brute force attack tool from Sensepost
    - WebScarab: web proxy tool from OWASP
    - Burp Proxy: web penetration testing tool from Portswigger
    - Backtrack 5: open-source penetration testing tool from www.ethical-Hacker.org