

Common Attack Pattern Enumeration and Classification — CAPEC™

A Community Knowledge Resource for Building Secure Software

CAPEC IS A PUBLICLY available catalog of attack patterns along with a comprehensive schema and classification taxonomy created to assist in the building of secure software. By standardizing the definition of attack patterns as part of a broader integrated software assurance knowledge architecture that includes similar knowledge-standardization efforts such as Common Weakness Enumeration (CWE), Common Vulnerabilities and Exposures (CVE), and Malware Attribute Enumeration and Characterization (MAEC), CAPEC supports the needs of developers, testers, and educators to build secure software and assists in enhancing security throughout the software development lifecycle.

Attack Patterns Examples:

- HTTP Response Splitting
- SQL Injection
- XSS in HTTP Query Strings
- Session Fixation
- Phishing
- Filter Failure through Buffer Overflow
- Removing or Short-Circuiting Guard Logic
- Lifting Data Embedded in Client Distributions
- Subvert Code-signing Facilities
- Reflection Attack in an Authentication Protocol
- Cause Web Server Misclassification
- Rainbow Table Password Cracking
- Forced Deadlock
- Cache Poisoning
- Restful Privilege Escalation

Challenge

As the tempo and complexity of building software increases and the number and skill level of attackers grow, ensuring an adequate level of security assurance becomes more challenging. Secure software builders must protect against potential vulnerabilities while software attackers often only require a single exposed vulnerability. The development community therefore needs more than just good software engineering and analytical practices, a solid grasp of software security features, and a powerful set of tools to identify and mitigate relevant vulnerabilities in software. To be effective, the community needs to think outside the box and have a firm grasp of the attacker's perspective and the approaches they use to exploit software.

Solution

Led by Cigital, Inc. and sponsored by the U.S. Department of Homeland Security's National Cyber Security Division as part of the Software Assurance strategic initiative, CAPEC makes the concept of attack patterns actionable for the broader community through:

- Standardizing the capture and description of attack patterns through definition of a standard schema.
- Collecting known attack patterns into an integrated enumeration that can be effectively leveraged, enhanced and expanded by the community.
- Classifying attack patterns such that users can easily identify the subset of the entire enumeration that is appropriate for their context.

Capturing attack patterns in such a formalized way can bring considerable value for software security considerations throughout all phases of the software development lifecycle and other security-related activities including: (1) Requirements Elicitation, (2) Architecture and Design, (3) Implementation and Coding, (4) Software Testing and Quality Assurance, (5) Systems Operation, and (6) Policy and Standard Generation.

Many other tools in addition to attack patterns are useful for building secure software such as misuse/abuse cases, security requirements, threat models, knowledge of common weaknesses and vulnerabilities, coding rules, and attack trees. However, attack patterns play a unique role amid this larger architecture of software security knowledge and techniques since an appropriate defense can only be established once you know how the software will be attacked.

Visit the CAPEC Web site to review source documents and the current draft of the CAPEC List.

Attack Patterns Are:

- Powerful mechanisms to capture and communicate the attacker's perspective.
- Descriptions of common methods for exploiting software.
- Derived from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples.



The MITRE Corporation—a Federally Funded Research and Development Center—maintains CAPEC and its public Web site and provides impartial technical guidance to the CAPEC Community throughout the process to ensure CAPEC serves the public interest.

202 Burlington Road, Bedford, MA 01730-1420
www.mitre.org

MITRE

Example Attack Pattern

Name	HTTP Response Splitting
Typical Severity	High
Description	<p>HTTP Response Splitting causes a vulnerable web server to respond to a maliciously crafted request by sending an HTTP response stream such that it gets interpreted as two separate responses instead of a single one. This is possible when user-controlled input is used unvalidated as part of the response headers. An attacker can have the victim interpret the injected header as being a response to a second dummy request, thereby causing the crafted contents to be displayed and possibly cached. To achieve HTTP Response Splitting on a vulnerable web server, the attacker:</p> <ol style="list-style-type: none"> 1. Identifies the user-controllable input that causes arbitrary HTTP header injection. 2. Crafts a malicious input consisting of data to terminate the original response and start a second response with headers controlled by the attacker. 3. Causes the victim to send two requests to the server. The first request consists of maliciously crafted input to be used as part of HTTP response headers and the second is a dummy request so that the victim interprets the split response as belonging to the second request.
Attack Prerequisites	<p>User-controlled input used as part of HTTP header</p> <p>Ability of attacker to inject custom strings in HTTP header</p> <p>Insufficient input validation in application to check for input sanity before using it as part of response header</p>
Typical Likelihood of Exploit	Medium
Methods of Attack	<p>Injection</p> <p>Protocol Manipulation</p>
Examples-Instances	In the PHP 5 session extension mechanism, a user-supplied session ID is sent back to the user within the Set-Cookie HTTP header. Since the contents of the user-supplied session ID are not validated, it is possible to inject arbitrary HTTP headers into the response body. This immediately enables HTTP Response Splitting by simply terminating the HTTP response header from within the session ID used in the Set-Cookie directive. CVE-2006-0207
Attacker Skill or Knowledge Required	High - The attacker needs to have a solid understanding of the HTTP protocol and HTTP headers and must be able to craft and inject requests to elicit the split responses.
Resources Required	None
Probing Techniques	<p>With available source code, the attacker can see whether user input is validated or not before being used as part of output. This can also be achieved with static code analysis tools</p> <p>If source code is not available, the attacker can try injecting a CR-LF sequence (usually encoded as %0d%0a in the input) and use a proxy such as Paros to observe the response. If the resulting injection causes an invalid request, the web server may also indicate the protocol error.</p>
Indicators-Warnings of Attack	The only indicators are multiple responses to a single request in the web logs. However, this is difficult to notice in the absence of an application filter proxy or a log analyzer. There are no indicators for the client
Solutions and Mitigations	To avoid HTTP Response Splitting, the application must not rely on user-controllable input to form part of its output response stream. Specifically, response splitting occurs due to injection of CR-LF sequences and additional headers. All data arriving from the user and being used as part of HTTP response headers must be subjected to strict validation that performs simple character-based as well as semantic filtering to strip it of malicious character sequences and headers.
Attack Motivation-Consequences	<p>Run Arbitrary Code</p> <p>Privilege Escalation</p>
Context Description	HTTP Response Splitting attacks take place where the server script embeds user-controllable data in HTTP response headers. This typically happens when the script embeds such data in the redirection URL of a redirection response (HTTP status code 3xx), or when the script embeds such data in a cookie value or name when the response sets a cookie. In the first case, the redirection URL is part of the Location HTTP response header, and in the cookie setting, the cookie name/value pair is part of the Set-Cookie HTTP response header.
Injection Vector	User-controllable input that forms part of output HTTP response headers
Payload	Encoded HTTP header and data separated by appropriate CR-LF sequences. The injected data must consist of legitimate and well-formed HTTP headers as well as required script to be included as HTML body.
Activation Zone	API calls in the application that set output response headers.
Payload Activation Impact	The impact of payload activation is that two distinct HTTP responses are issued to the target, which interprets the first as response to a supposedly valid request and the second, which causes the actual attack, to be a response to a second dummy request issued by the attacker.
Related Weaknesses	<p>CWE113 - HTTP Response Splitting - Targeted</p> <p>CWE74 - Injection - Secondary</p>
Relevant Security Requirements	All client-supplied input must be validated through filtering and all output must be properly escaped.
Related Security Principles	Reluctance to Trust
Related Guidelines	Never trust user-supplied input.
References	G. Hoglund and G. McGraw. Exploiting Software: How to Break Code. Addison-Wesley, February 2004.