Adapted from Software Security Engineering: *A guide for Project Managers* by Julia Allen, Sean Barnum, Robert Ellison, Gary McGraw, Nancy Mead

Pattern name and classification	A unique, descriptive identifier for the pattern
Attack prerequisites	Which conditions must exist or which functionality and which characteristics must the target software have, or which behavior must it exhibit, for this attack to succeed?
Description	A description of the attack, including the chain of actions taken.
Related vulnerabilities or weaknesses	Which specific vulnerabilities or weaknesses does this attack leverage? Specific vulnerabilities should reference industry-standard identifiers such as common vulnerabilities and exposures (CVE) number or USE-CERT number. Specific weaknesses (underlying issues that may cause vulnerabilities) should reference industry-standard identifiers such as Common Weaknesses Enumeration (CWE).
Method of attack	What is the vector of attack used (e.g., malicious data entry, maliciously crafted file, protocol corruption)?
Attack motivation-- consequences	What is the attacker trying to achieve by using this attack? This is not the end business/mission goal of the attack within the target context, but rather the specific technical result desired that could be used to achieve the end business/mission objective. This information is useful for aligning attack patterns to threat models and for determining which attack patterns from the broader set available are relevant for a given context.
Attacker skill or knowledge required	What level of skill or specific knowledge must the attacker have to execute such an attack? This should be communicated on a rough scale (e.g., low, moderate, high) as well as in contextual detail of which type of skills or knowledge are required.
Resources required	Which resources (e.g., CPU cycles, IP address, tools, time) are required to execute the attack?

Solutions and mitigations	Which actions or approaches are recommended to mitigate this attack, either through resistance or through resiliency?
Context description	In which technical contexts (e.g., platform, operating system, language, and architectural paradigm) is this pattern relevant? This information is useful for selecting a set of attack patterns that are appropriate for a given context.
References	What other sources of information are available to describe this attack.

Example of an attack Pattern

Pattern name and classification	Make a client invisible
Attack prerequisites	The application must have a multi-tiered architecture with a division between the client and the server.
Description	This attack pattern exploits client-side trust issues that are apparent in the software architecture. The attacker removes the client from the communication loop by communicating directly with the server. This could be done by bypassing the client or by creating a malicious impersonation of the client.
Related vulnerabilities or weaknesses	Man-in-the-Middle (MITM)(CWE #300), Origin Validation Error (CWE #346), Authentication Bypass by Spoofing (CWE #290), No Authentication for Critical Function (CWE #306), Reflection Attack in an Authentication Protocol (CWE #301).
Method of attack	Direct protocol communication with the server
Attack motivation-consequences	Potentially information leak, data modification. Arbitrary code execution and so on. These can all be achieved by bypassing authentication and filtering accomplished with this attack pattern.

Attacker skill or knowledge required	Finding and initially executing this attack requires a moderate skill level and knowledge of client/server communications protocol. Once the vulnerability is found, the attack can be easily automated for execution by far less skilled attackers. Skill levels for follow-on attacks can vary widely depending on the nature of the attack.
Resources required	None, although protocol analysis tools and client impersonation tools such as netcat can greatly increase the ease and effectiveness of the attack.
Solutions and mitigations	Increase attack resistance. Use strong two-way authentication for all communication between the client and the server. This option could have significant performance implications. Increase attack resilience: Minimize the amount of logic and filtering present on the client; place it on the server instead. Use white lists on the server to filter and validate client input.
Context description	“Any raw data that exist outside the server software cannot and should not be trusted. Client-side security is an oxymoron. Simply put, all clients will be hacked. Of course, the real problem is one of client-side trust. Accepting anything blindly from the client and trusting it through and through is a bad idea, and yet this often the case in server-side design.”
Reference	*Exploiting Software: How to Break Code, p.150*