<UWE-484-03>: <Tinyserv Authentication Session Fixation

>

The Basics

Authors: Lanting Lu, Huijie Li
Disclosure Date: 03/16/2025
Product: tinyserv
Reporter(s): CSE 484 staff
Exploit sample: see sploit3.sh

The Exploit

Exploit strategy (2p) - What does the exploit do? What are the key parts of the exploit?:
The exploit script aims to access a protected admin.txt file sending a crafted HTTP GET request. It utilizes a specially generated cookie to bypass normal authentication checks, potentially exploiting weaknesses in how the server validates cookies.
This exploit abuses a session fixation flaw in Tinyserv's authentication system—session cookies aren't properly handled or invalidated. Here's how it works: The attacker first logs in as an admin and accesses admin.txt to establish an active session. The sploit4.sh script then hijacks that same session cookie to send unauthorized requests (via curl) to collideme41. Since Tinyserv doesn't enforce session expiration or ensure unique sessions, the attacker bypasses re-authentication entirely and grabs restricted data. Essentially, they piggyback on the admin's valid session to read protected files they shouldn't have access to.

Exploit primitives (2p) - What new capabilities (if any) does an attacker gain from successful exploitation of the vulnerability? These are small, process-space or environment-related building blocks used to construct full exploit chains such as: write arbitrary data to arbitrary memory locations, overwrite local variables, overwrite heap data, crash the program, send data on the network, read from local files, escalate privileges, etc.:
This flaw lets attackers keep reusing active sessions to stay in protected zones without ever logging in again. Once they snag a valid session token, it stays valid forever—so they can grab files like admin.txt anytime. Attackers can script this exploit to run automatically, letting them quietly maintain access for weeks or months. Pair this with something like session hijacking, and you've got a recipe for total admin account takeover—no password needed.

The Vulnerability

Vulnerability details (3p) - Provide details about the vulnerability. Also, if applicable, provide the vulnerable file(s), function(s), and line number(s):

This flaw boils down to Tinyserv's broken session token management—old tokens never truly die. The system doesn't enforce session expiration, unique tokens, or even kill sessions when users log out. Attackers can simply grab a valid token once (like from a leaked session) and reuse it forever to hijack admin privileges. Since Tinyserv doesn't check if sessions are fresh or tie them to specific users/IPs, attackers can keep riding the same token like a skeleton key, bypassing login checks entirely. It's like leaving a master key under the doormat and never changing the locks.

Bug class (CWE) (1p) - Visit this page (https://cwe.mitre.org/data/definitions/699.html) and provide the name and CWE identifier number of the bug class that best fits the vulnerability. You will need to expand the dropdowns/bullets to see all the options for each category.:
CWE524 Use of Cache Containing Sensitive Information
/CWE281 Improper Preservation of Permissions

Severity/priority score (2p) - Rate the severity of the vulnerability as one of low, medium, or high, and explain your choice in terms of the vulnerability and the provided exploit:
We assess the severity of this vulnerability as High, as it allows attackers to access sensitive information in admin.txt without authentication. This vulnerability falls under CWE-524 (Use of Cache Containing Sensitive Information) and CWE-281 (Improper Preservation of Permissions). The core issue is that Tinyserv's caching mechanism does not properly distinguish between regular files and sensitive files. Attackers can exploit this by simulating a legitimate user accessing admin.txt, causing the server to cache its contents. Once cached, an attacker can directly request the same file and retrieve the cached content, bypassing authentication entirely. This exploit requires minimal technical skill and can lead to serious consequences such as account compromise, privilege escalation, or unauthorized data access. Given the low complexity of exploitation and the potential high impact on security, we classify the severity of this vulnerability as High.

Thoughts on how this vulnerability might have been found by the attacker (1p) - What did the attacker do (what tools, techniques, etc.) to find this vulnerability?

The Patch

Proposed patch plan (2p) - This must be something you can implement in code for tinyserv; do not say something like "use stack canaries" for your answer to this question:

Our patch plan aims to address the vulnerability in Tinyserv's caching mechanism, which currently allows attackers to access sensitive files like admin.txt without authentication. This issue arises because Tinyserv does not differentiate between regular and sensitive files when caching. Once an administrator accesses admin.txt, the server stores its content in the cache. Attackers can then make a request for the same file and retrieve the cached content, bypassing authentication entirely.
To mitigate this, we plan to implement a function called sensitive_file(), which will identify

whether a requested file is sensitive. This function will check if the file path contains admin.txt before the server caches the response. If the function detects a sensitive file, the server will skip caching it entirely, preventing unauthorized users from accessing previously stored admin data. This change will be integrated before calling update_cache(), ensuring that only non-sensitive files are cached while confidential files remain protected. By applying this fix, Tinyserv will maintain its caching functionality for regular files while blocking attackers from exploiting cached admin content.

Killing the bug class (bonus 1p) - What could the tinyserv project do to prevent more vulnerabilities of this CWE type from appearing anywhere in this codebase in the future? Explain some advantages and disadvantages of your solution: