ETX Signal-X

Daily Intelligence Digest

Saturday, February 28, 2026

9	15
ARTICLES	TECHNIQUES

Article 1

OS Command Injection—When Your Server Obeys My Commands Like a Soldier

Source: securitycipher

Date: 28-Jun-2025

URL: https://infosecwriteups.com/os-command-injection-when-your-server-obeys-my-commands-like-a-soldier-ecbe2fe6ec3d

⚠️ No actionable techniques found

Article 2

Cracking the Clock: How I Took Over Any Account Using a Timestamp Leak

Source: securitycipher

Date: 11-Jun-2025

URL: https://ritikver22000.medium.com/cracking-the-clock-how-i-took-over-any-account-using-a-timestamp-leak-516fc88c0113

T1 Predictable Password Reset Token via Timestamp Leak

💉 Payload

```
victim@example.com timestamp @example timestamp
```

⚔️ Attack Chain

1. Initiate a password reset for a target user's email (e.g., victim@example.com) and intercept the server's response.
2. Observe the response headers to determine the current server time.
3. Subtract 2 seconds from the server response time to match the timestamp format used in the reset token.
4. Concatenate the target email and the calculated timestamp in the format: `victim@example.com timestamp @example timestamp`.
5. Base64-encode the resulting string to forge a valid reset token.
6. Use the forged token in the password reset link to reset the victim's password and gain account access.

🔍 Discovery

While testing the password reset flow, the researcher noticed that the Base64-encoded reset token, when decoded, revealed the user's email and a timestamp. By comparing multiple requests, it was observed that the token's timestamp was always 2 seconds behind the server's response time header, indicating predictability.

🔓 Bypass

The attack bypasses any randomness or entropy in the password reset process by exploiting the deterministic relationship between the server's response time and the token's timestamp. No confirmation emails or secondary verification steps were present, enabling direct account takeover.

🔗 Chain With

Can be chained with email enumeration or user ID discovery to automate mass account takeovers. May enable further privilege escalation if the reset process grants elevated access or triggers additional flows (e.g., session fixation, forced password change on admin accounts).

Article 3

The Never-Ending Party: Invite Links That Never Die

Source: securitycipher

Date: 08-Sep-2025

URL: https://ch1ta.medium.com/the-never-ending-party-invite-links-that-never-die-a6b000901477

T1 Persistent, Non-Expiring Invite Token Abuse

💉 Payload

```
[Invite Link 1: Admin]
https://target.ai/invite?token=ADMIN_TOKEN

[Invite Link 2: User]
https://target.ai/invite?token=USER_TOKEN
```

⚔️ Attack Chain

1. Attacker receives two invite links to the same email address, each with a different role (e.g., Admin and User).
2. Attacker accepts the User invite link and joins the organization with basic privileges.
3. Attacker is later removed from the organization by an admin.
4. Attacker reuses the previously issued Admin invite link (still valid) to rejoin the organization with elevated privileges (Admin access).

🔍 Discovery

Manual exploration of the invite system, specifically testing invite link validity after user removal and after multiple invites to the same email with different roles.

🔓 Bypass

Invite tokens are not invalidated after use, user removal, or new invite issuance. This allows previously removed or downgraded users to regain access or escalate privileges by reusing old invite links.

🔗 Chain With

Combine with lateral movement or privilege escalation attacks post-rejoin. Use for persistent access or as a foothold for further organizational compromise.

T2 Simultaneous Multi-Role Invite Link Exploitation

💉 Payload

```
[Invite Link 1: Admin]
https://target.ai/invite?token=ADMIN_TOKEN

[Invite Link 2: User]
https://target.ai/invite?token=USER_TOKEN
```

⚔️ Attack Chain

1. Attacker receives multiple invites to the same email address, each with a different role (e.g., Admin and User).
2. Attacker accepts the User invite to join with basic privileges.
3. At any later time, attacker uses the Admin invite link (still valid) to escalate privileges to Admin without any additional approval or notification.

🔍 Discovery

Tested the effect of accepting multiple invites with different roles to the same email and observed that all links remained valid and could be used independently at any time.

🔓 Bypass

The system fails to invalidate older or lower-privilege invite tokens after one is used, allowing privilege escalation by reusing higher-privilege invites.

🔗 Chain With

Use for stealth escalation after initial low-privilege access. Chain with social engineering to increase likelihood of multiple role invites.

T3 Rejoining After Removal via Stale Invite Link

💉 Payload

```
[Invite Link: Any valid invite issued prior to removal]
https://target.ai/invite?token=ANY_VALID_TOKEN
```

⚔️ Attack Chain

1. Attacker is invited and joins the organization using a valid invite link.
2. Attacker is removed from the organization by an admin.
3. Attacker reuses the original invite link (which remains valid) to rejoin the organization without admin approval.

🔍 Discovery

Tested the validity of invite links after user removal and confirmed that removed users could rejoin without new approval.

🔓 Bypass

Invite tokens are not revoked upon user removal, allowing persistent unauthorized re-entry.

🔗 Chain With

Use for persistent access after detection and removal. Combine with privilege escalation if multiple role invites exist.

Article 4

Bypassing methods that I used to find CSRF vulnerabilities

Source: securitycipher

Date: 22-Aug-2024

URL: https://anonysm.medium.com/bypassing-methods-that-i-used-to-find-csrf-vulnerabilities-b7dbf88cdb0a

T1 CSRF Token Parameter Removal or Blank Value

💉 Payload

```
POST /password_change
Host: email.example.com
Cookie: session_cookie=YOUR_SESSION_COOKIE

new_password=abc123
```

⚔️ Attack Chain

1. Identify a CSRF-protected endpoint (e.g., /password_change).
2. Craft a POST request omitting the csrf_token parameter entirely, or include it with a blank value.
3. Submit the request using a crafted HTML form or direct HTTP request.
4. Observe if the action is executed without CSRF validation.

🔍 Discovery

Manual testing of endpoints with and without the CSRF token parameter, and with a blank value, to identify logic flaws in token validation.

🔓 Bypass

Some applications only validate the CSRF token if it is present and non-blank. If missing or blank, validation is skipped and the action is executed.

🔗 Chain With

Can be combined with other parameter pollution or request smuggling techniques if the application parses parameters inconsistently.

T2 Using Attacker's Own Valid CSRF Token

💉 Payload

```
POST /password_change
Host: email.example.com
Cookie: session_cookie=YOUR_SESSION_COOKIE

new_password=abc123&csrf_token=YOUR_TOKEN
```

⚔️ Attack Chain

1. Obtain a valid CSRF token from your own (attacker) session or a test account.
2. Craft a POST request to the target endpoint using the attacker's CSRF token.
3. Submit the request as the victim (e.g., via CSRF HTML form or XHR).
4. If the application only checks token validity (not session binding), the action is executed for the victim.

🔍 Discovery

Testing endpoints by submitting requests with a valid CSRF token from a different session/user.

🔓 Bypass

Application checks if the token is valid but does not verify it is bound to the current user's session.

🔗 Chain With

Can be chained with session fixation or account takeover if session tokens are also predictable or shared.

T3 Bypassing Referer Header CSRF Protections

💉 Payload

````
Remove Referer:
```
<html>
<meta name="referrer" content="no-referrer">
<form method="POST" action="https://email.example.com/password_change" id="csrf-form">
<input type="text" name="new_password" value="abc123">
<input type='submit' value="Submit">
</form>
<script>document.getElementById("csrf-form").submit();</script>
</html>
```

Referer as subdomain:
```
POST /password_change
Host: email.example.com
Cookie: session_cookie=YOUR_SESSION_COOKIE;
Referer: example.com.attacker.com

new_password=abc123
```

Referer as path:
```
POST /password_change
Host: email.example.com
Cookie: session_cookie=YOUR_SESSION_COOKIE;
Referer: attacker.com/example.com

new_password=abc123
```
````

⚔️ Attack Chain

1. Identify endpoints protected by Referer header checks instead of CSRF tokens.
2. Attempt to remove the Referer header by using a <meta name="referrer" content="no-referrer"> tag in the attacker's HTML page.
3. If removal fails, craft requests with Referer header containing the target domain as a subdomain or path (e.g., example.com.attacker.com or attacker.com/example.com).
4. Submit the request and observe if the action is executed.

🔍 Discovery

Manual testing of Referer header manipulation and observing application logic for domain substring checks or presence-only validation.

🔓 Bypass

Application only validates Referer if present, so omitting it bypasses the check. Application uses substring matching (e.g., "example.com" in Referer), allowing crafted subdomains or paths to pass validation.

🔗 Chain With

Can be chained with open redirect or reflected URL injection if the Referer is user-controlled elsewhere.

T4 Changing HTTP Request Method to Bypass CSRF Protections

💉 Payload

```
GET /password_change?new_password=abc123
Host: email.example.com
Cookie: session_cookie=YOUR_SESSION_COOKIE
```

⚔️ Attack Chain

1. Identify endpoints that accept multiple HTTP methods (e.g., both POST and GET).
2. Attempt the state-changing action using an alternate method (e.g., GET instead of POST) without CSRF token.
3. Use an HTML element (e.g., <img src=...>) to trigger the request from the victim's browser.
4. Observe if the action is executed without CSRF validation.

🔍 Discovery

Testing endpoints with alternate HTTP methods to see if CSRF protection is only enforced on certain methods.

🔓 Bypass

Application enforces CSRF protection only on specific HTTP methods (e.g., POST), but not on others (e.g., GET).

🔗 Chain With

Can be chained with CORS misconfigurations or open redirects if the endpoint is also accessible cross-origin.

Article 5

How did I found Account Takeover Vulnerability on takeuforward.org

Source: securitycipher

Date: 15-Nov-2024

URL: https://rajukani100.medium.com/how-did-i-found-account-takeover-vulnerability-on-takeuforward-org-735630b4167c

T1 Client-Side OTP Hash Manipulation for Account Takeover

💉 Payload

```
Cookie: otp_key=$2a$12$VI1/nmpcWUkeku8p63QIDe2lABr1S39U.f0GO0As.4hHfhU7dloz4
```

⚔️ Attack Chain

1. Initiate a password reset for the target account via the "Forgot Password" functionality.
2. Receive an OTP on the target's email (not needed for the exploit).
3. On the OTP verification page, generate a bcrypt hash for any arbitrary OTP value (e.g., 569656) using cost factor 12.
4. Replace the value of the `otp_key` cookie in the browser with the attacker-generated bcrypt hash.
5. Submit the password reset form with the arbitrary OTP and a new password.
6. The server validates the attacker-controlled hash, allowing password reset and full account takeover.

🔍 Discovery

While intercepting the password reset flow with Burp Suite, the researcher noticed an `otp_key` cookie containing a bcrypt hash. By hypothesizing that the server compared the submitted OTP to the hash in the cookie (potentially client-side or with no server-side validation), the researcher tested replacing the hash with their own and confirmed the bypass.

🔓 Bypass

The server trusts the client-supplied `otp_key` hash for OTP verification, allowing attackers to supply their own hash for any OTP value, bypassing the need for the real OTP.

🔗 Chain With

Combine with email enumeration to automate mass account takeovers. Use in conjunction with session fixation if the platform issues new sessions post-reset. Potential for privilege escalation if admin accounts are targeted.

Article 6

Blind SSRF with Out-of-Band Detection: Step-by-Step Exploitation & Prevention — SSRF Labs

Source: securitycipher

Date: 03-Feb-2025

URL: https://bashoverflow.medium.com/blind-ssrf-with-out-of-band-detection-step-by-step-exploitation-prevention-ssrf-labs-d8a4d890184d

T1 Blind SSRF via Manipulated Referer Header with Out-of-Band Detection

💉 Payload

```
GET /product?productId=1 HTTP/1.1
Host: targetsite.com
Referer: http://<your-burp-collaborator-id>.burpcollaborator.net/
[Other headers as needed]
```

⚔️ Attack Chain

1. Identify a product page request (e.g., `GET /product?productId=1`) in the application.
2. Send the request to Burp Suite's Repeater tab.
3. Replace the `Referer` header value with a unique Burp Collaborator URL (e.g., `http://<your-burp-collaborator-id>.burpcollaborator.net/`).
4. Send the modified request to the server.
5. Monitor the Burp Collaborator tab for any HTTP or DNS interactions.
6. If an interaction is observed, confirm that the server made an out-of-band request, indicating a blind SSRF vulnerability.

🔍 Discovery

Observed that the application fetches a URL from the `Referer` header for analytics purposes. Hypothesized SSRF by manipulating the header and confirmed via out-of-band interaction using Burp Collaborator.

🔓 Bypass

No explicit bypass logic described, but the use of the `Referer` header (often overlooked in SSRF filtering) is itself a bypass technique compared to more commonly filtered parameters.

🔗 Chain With

Can be chained with internal IP/host enumeration by replacing the Collaborator URL with internal addresses (e.g., `http://169.254.169.254/latest/meta-data/`). Potential to escalate to RCE if internal services are exposed or if SSRF leads to further vulnerabilities (e.g., accessing admin panels, cloud metadata endpoints).

Article 7

Business Logic Flaw worth $1250

Source: securitycipher

Date: 19-Apr-2025

URL: https://vijetareigns.medium.com/business-logic-flaw-worth-1250-35efcd1b9af9

T1 Forced Browsing of Email Verification Endpoint Enables Account Takeover

💉 Payload

```
GET https://app.redacted.tv/verify?email=user@gmail.com
```

⚔️ Attack Chain

1. Register a new account on the target crypto wallet platform using the victim's email address.
2. Wait for the verification email to be sent to the victim (or use a controlled email for testing).
3. Copy the verification URL (e.g., `https://app.redacted.tv/verify?email=user@gmail.com`).
4. Open the verification URL in an incognito browser or a different browser session where no authentication cookies are present.
5. Observe that the verification page loads without requiring a password, and prompts for the verification code.
6. If the attacker can obtain or guess the verification code (e.g., via email interception, brute-force, or social engineering), they can complete the verification and take over the account.

🔍 Discovery

Noticed that the email verification URL could be accessed in an unauthenticated browser session (incognito), which loaded the verification page without requiring prior authentication or password entry.

🔓 Bypass

The endpoint fails to enforce session or authentication checks before allowing access to the verification page, relying solely on possession of the verification code.

🔗 Chain With

Can be chained with email interception, phishing, or brute-forcing the verification code to achieve full account takeover. If 2FA setup is not enforced or can be bypassed, this could lead to persistent access.

Article 8

Insecure Direct Object Reference (IDOR) in engcastleportal.com

Source: securitycipher

Date: 27-Oct-2025

URL: https://medium.com/@mohammedmogeab/insecure-direct-object-reference-idor-in-engcastleportal-com-a2ac44d62f00

T1 Mass Student PII Enumeration via IDOR on Profile Edit Endpoint

💉 Payload

```
GET /student/profile/{user_id}/edit HTTP/1.1
Host: engcastleportal.com
Cookie: engcastle_session=<valid_session_cookie>
```

⚔️ Attack Chain

1. Authenticate as any valid student user on https://engcastleportal.com.
2. Capture the session cookie (e.g., `engcastle_session`).
3. Send GET requests to `/student/profile/{user_id}/edit` incrementing the `user_id` integer (e.g., 3501, 3502, ...).
4. Each request returns the full profile form of the target student, exposing PII: first name, last name, email, phone number, date of birth.
5. Automate the process to scrape thousands of student records.

🔍 Discovery

Manual review of profile edit endpoint revealed lack of authorization checks on the `user_id` path parameter. Incrementing the ID in the URL returned other users' data.

🔓 Bypass

No server-side validation that the requested `user_id` matches the authenticated session. Any authenticated user can enumerate all student profiles by manipulating the path parameter.

🔗 Chain With

Use harvested PII (emails, names, phone numbers) for downstream attacks: phishing, identity theft, social engineering. Enables mass data scraping for credential stuffing or further enumeration.

T2 Full Account Takeover via Email Enumeration and Weak Authentication/Reset Logic

💉 Payload

```
GET /student/profile/{user_id}/edit HTTP/1.1
Host: engcastleportal.com
Cookie: engcastle_session=<valid_session_cookie>
```

⚔️ Attack Chain

1. Use TECHNIQUE 1 to enumerate student emails and user_ids via the IDOR endpoint.
2. Identify the platform's authentication or password reset mechanism (e.g., via `/reset-password` or login form).
3. Attempt password reset for a target email. If the reset flow lacks strict verification (e.g., no email confirmation, token, or CAPTCHA), reset the password using only the email address.
4. Alternatively, test for weak default password schemes (e.g., password derived from user_id or email).
5. Log in as the victim student, achieving full account takeover.

🔍 Discovery

Analysis of platform behavior after PII enumeration suggested authentication logic was tied to predictable identifiers (user_id, email). Testing password reset and login flows revealed insufficient verification.

🔓 Bypass

If password reset requires only email (no token/confirmation), attacker can reset any account using enumerated emails. If credentials are derived from predictable patterns (user_id, email), attacker can guess or brute-force passwords.

🔗 Chain With

Combine with TECHNIQUE 1 for horizontal privilege escalation across all student accounts. Persistent access to internal features, bypassing all privacy controls. Enables automation for platform-wide account compromise.

Article 9

From Path Guessing to Dashboard Takeover: Full Access to Government Data via Broken Access Control

Source: securitycipher

Date: 30-Jul-2025

URL: https://0xhamod.medium.com/from-path-guessing-to-dashboard-takeover-full-access-to-government-data-via-broken-access-control-a4c048fc05bb

T1 Unauthenticated Access to Country-Specific Update Dashboards via Path Guessing

💉 Payload

```
GET /update_newzealand_indicators.php HTTP/1.1
Host: <target>
```

⚔️ Attack Chain

1. Identify the pattern of country-specific login endpoints (e.g., /login_newzealand_indicators.php) from the application structure or homepage.
2. Replace the "login" prefix with "update" to form a new endpoint (e.g., /update_newzealand_indicators.php).
3. Send a direct unauthenticated GET request to the crafted update endpoint.
4. Gain full access to the dashboard, including the ability to edit indicators, upload files, and save changes without authentication.

🔍 Discovery

Pattern recognition of endpoint naming conventions during manual review of country-specific login pages. The researcher hypothesized a parallel update endpoint by substituting the prefix and tested it directly.

🔓 Bypass

No authentication, session, token, or CSRF protection required; direct access is possible simply by knowing or guessing the endpoint path.

🔗 Chain With

Use with wordlists of country names to enumerate and access multiple dashboards (horizontal privilege escalation). Potential to upload malicious files or manipulate sensitive data across multiple country dashboards.

T2 Client-Side JavaScript Disclosure of Sensitive Update Endpoints

💉 Payload

```
// In a public .js file:
var updateUrl = "/update_newzealand_indicators.php";
```

⚔️ Attack Chain

1. Review all loaded JavaScript files for client-side references to sensitive endpoints.
2. Identify explicit references to update endpoints (e.g., /update_newzealand_indicators.php) in JavaScript code.
3. Use the discovered endpoint to send unauthenticated requests, exploiting the lack of access control.

🔍 Discovery

Manual inspection of JavaScript files for hidden or undocumented API endpoints, revealing unauthenticated update URLs.

🔓 Bypass

Endpoint is exposed client-side and lacks any session, token, or CSRF protection, allowing direct exploitation by anyone with the URL.

🔗 Chain With

Combine with endpoint fuzzing to automate discovery of additional unauthenticated update endpoints. Use client-side leaks to inform further enumeration or privilege escalation.