

Other Web Application Vulnerabilities

Connor Corcoran

Lab Setup

Environment: Azure VM running Windows Server with IIS hosting + PfSense running as the network firewall

Files Deployed: addalbum.aspx, listalbums.aspx, upload.aspx, login.aspx, firectiorylist.aspx, plus helper files (getcookies.htm, logit.aspx, hair.svg).

Permissions & config: wwwroot/uploads writable by IIS_IURS; web.config modified to allow HTML input

Web Application Threats

Cross Site Scripting

What: Occurs when an application displays user input on a page allowing injected Javascript or HTML to run in a users browser

Impact: Can lead to theft of session cookies, user impersonation, or unauthorized actions

Fix: Encode output

Detection: Review logs for missing or invalid CSRF tokens and unexpected state changes

Cross Site Request Forgery

What: Tricks and authenticated users browsers into sending unintended requests to a web application

Impact: May cause unauthorized changes such as role updates or data modifications

Fix: implement anti-CSRF tokens, use POST for sensitive actions, and set cookies with SameSite protection

Detection: Review logs for missing or invalid CSRF tokens and unexpected state changes.

Session Hijacking

What: attackers obtain a valid session token and refuses it to impersonate a user

Impact: Account takeover - access/modify sensitive data, perform actions.

Fix: Use HttpOnly Secure, SameSite cookies; enforce HTTPS/HSTS; regenerate session id on login/privilege change; short timeouts

Detection: monitor same session ID from multiple IPs/UA; check cookies in browser DevTools

Unrestricted File Upload

What: App accepts and serves uploaded files without validating type/contents or blocking execution.

Impact: Upload webshells or HTML/JS pages -> remote code execution, data exposure, persistent backdoors.

Fix: Store uploads outside webroot or block script handlers in upload folder.

Detection: Alert on executable/disallowed uploads and access to uploaded files

Cross-Site Scripting (XSS): Attacker Impact

Steal or reuse session cookies (if not HttpOnly), carry out admin actions.

Deface site or insert malicious iframes/phishing pages.

Run actions in context of user.

Data exfiltration from pages the user can access.

XSS: where it arises in your code

The image consists of two screenshots. The top screenshot shows a browser window with an error message: "A potentially dangerous Request.QueryString value was detected from the client (<0>=<0>T0le101</0>)". The bottom screenshot shows a file editor displaying a portion of a web.config file.

```
<httpRuntime validateRequest="true" requestValidationMode="2.0" />
```

Where it arises

- Origin: add album.aspx (user enters album title/artist/description) → saved to database or storage.
- Enabling config: web.config set to allow HTML (e.g. pages validateRequest= "false" or requestValidationMode= "2.0") which lets <script>/<iframe> to pass through

Why the code is vulnerable

- The app renders untrusted input directly into HTML
- The web.config change removed an input level filter so malicious markup can be saved

XSS: Secure Fixes

Cross-Site Scripting Fixes

- 01 Escape user input
- 02 Sanitize user input
- 03 Validate input
- 04 Use a web application firewall (WAF)
- 05 Implement content security policy (CSP)



Implement a multi-layered approach using input validation (sanitizing and filtering user input) and output encoding (correctly escaping data before displaying it)

Use Content Security Policy headers to instruct browser on what resources are safe to load.

Set the HttpOnly flag for cookies

Unrestricted File Upload: Attacker Impact

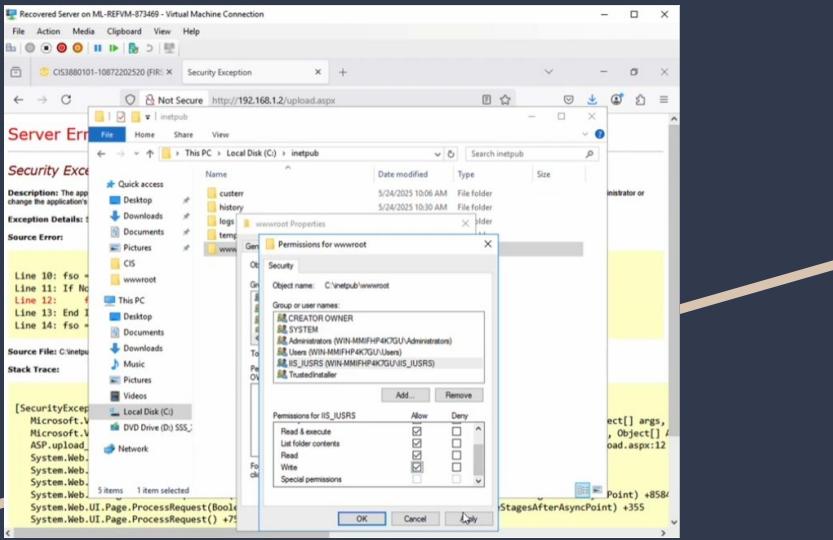
Upload and run a webshell/backdoor on the server.

Read or steal sensitive files (configs, source, DB creds).

Maintain persistent access (hidden/backdoor files).

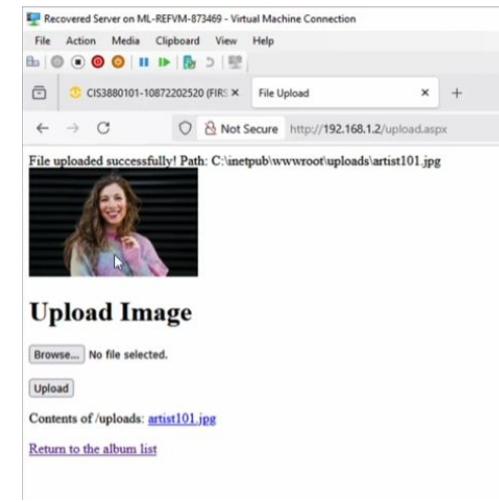
Deploy malware or disrupt service (ransomware/defacement).

Unrestricted File Upload: Where it Arises



Where it arises:

- Upload.aspx – file input saved to ~/uploads under wwwroot using use supplied filename (e.g. SaveAS(Server.MapPath("~/uploads/" + fileName)))
- Also granted IIS_IUSRS write permission to uploads folder so uploads succeed



Unrestricted File Upload: Fixes



1. Whitelist file extension and check content by magic bytes
2. Rename files to same names and store outside wwwroot:
3. Block execution in /uploads if inside web root by policing a web.config that denies .aspx
4. Tighten ACLs: minimal write permission for app account only; deny execute where possible.
5. Scan uploads with AV/malware scanner and limit file sizes

Cross-Site Request Forgery (CSRF): Attacker Impact

Attackers crafts a page that triggers the admins browser to call a “promote” URL while admin is authenticated → silent privilege escalation

Consequence: attacker can make an admin change other employees to admin, pivot, steal PII, etc.

CSRF: where it arises

Welcome, Alyssa Sprinkle				
Employees list				
123 Bookie Vacca	7773 Shady Orchard	7031969901 8044271243	grahab@icloud.com	Promote
222 Border Castor	6512 Iron Mall	5409921586 571640496	cgughaves@verizon.net	Promote
241 Alyssa Sprinkle	617 Red Frost	2769790609 5716513330	polices@adl.org	Promote
291 Cassie Dade	3067 Grand Forest Path	57113121069 8048159984	hamilton@live.com	Promote
293 Mitch Lemaster	981 Lary Rise By-pass	4342614746 5408441379	dmattt@sbcglobal.net	Promote
580 Mandi Markley	8588 Devy Pioneer Cape	7039559448 5409432066	calmous@hotmail.com	Promote
594 Vista Hayvenny	1386 Old Boulevard	5407804067 7039089072	mlasoum@verizon.net	Promote
601 Alicia Sharples	9627 Rustic Bluff Village	7034761501 2767101807	lspetres@sbcglobal.net	Promote
611 Robbi Finnbrown	1881 Colonial Wagon Hollow	8043611840 7574254331	drew@att.net	Promote
613 Teddy Abrahams	2562 Harry Quail Concessions	7035118224 434171916	matty@comcast.net	Promote
623 Khalilah Rousseen	3872 Gentle Willow Green	5714288360 804469482	mrdrivz@aol.com	Promote
635 Innis Sokolowski	3670 Noble Leaf Dell	5404809608 757806513	claypool@yahoo.com	Promote
637 Sel Mcconville	9999 Lost Dale	5404809608 757806513	lmartha@mac.com	Promote
642 Jettie Sweetland	4669 Spring Timber Lane	5401655391 524745079	bschulz@bellsouth.net	Promote
645 Clelia Chantrell	7623 Quashan Butterfly Forest	7037451406 8041912234	alephonsus@bluehost.com	Promote
655 Bennett Sanchez	3613 Little Creek	5404867471 7576154409	richard@live.com	Promote
665 Noemi Yorke	1881 Merry Grove	5710909217 8047135863	gryne@verizon.net	Promote
674 A. Conner	8118 Golden Mountain Range	6343218675 5404273232	ambuck@gmx.com	Promote
684 Pamela Mowry	9680 Harvest Rue	7576282808 2762617065	oplumus@mrex.com	Promote
685 Heidi Pound	2499 Stoney Prairie Passage	7032840839 5714352348	woodles2@gmail.com	Promote
705 Terence Ketsos	9266 Misty Crescent	4344449920 5711491407	fglock@att.net	Promote
713 Elda Furtado	1519 Sunny Zephyr Via	7572453404 7030344774	mschoffman@live.com	Promote
722 Eula Espinoza	8362 Bright Wind	7578792358 804069733	maricophil@icloud.com	Promote
724 Ruberto McCollum	5535 Broad Nectar Division	7172007448 7038420866	awinvite@hotmail.com	Promote
729 Adelina Coben	4198 Derry Embers Campus	6045309201 757327289	franelx@gmail.com	Promote

Where it arises:

- Endpoint: login.aspx or whatever page performs the promote action.
- The promote/promotion logic executes from a GET request rather than requiring a POST and token

Why the code is vulnerable:

- State-changing actions are allowed via GET and without any proof that the request originated from the app user interface
- Browser automatically attaches cookies to cross-origin resource loads, so the server cannot tell the request was forged

CSRF Fixes



Require POST for state changes.

Add anti-CSRF token: The server generates a unique unpredictable token for each user session.

Use SameSite cookies: Controls when cookies are sent with cross-site requests.

Session Hijacking: Attacker Impact

Immediately impersonate the victim (account takeover).

Access privileged functions if the victim is an admin.

Exfiltrate confidential data accessible to the user.

Perform fraudulent actions that appear legitimate.

Session Hijacking: where it arises

The screenshot shows a browser window with the URL `192.168.1.2/login.aspx`. On the left, there is a table titled "Employee list" displaying employee information. On the right, the developer tools' Network tab is open, showing a list of resources. One resource, "Cookies", is expanded, showing a table with columns "Name", "Value", and "SessionID=...". The "Value" column contains session IDs for various users. Below the table, it says "No cookie selected" and "Select a cookie to preview its value".

ID	Name	Address	Phone	Ext.
120	Booker Vacca	7773 Shady Orchard	7031969901	804427124
227	Bardie Castor	6512 Iron Mall	5409921586	571640408
247	Alyssa Sprinkle	617 Red Front	2769790609	571651333
291	Cassey Dade	3067 Grand Forest Path	5713121069	804819599
295	Mitch Lemaster	981 Lazy Rise By-pass	4342614746	540494433
580	Mundi Markley	8888 Dewy Pioneer Cape	7039559444	540944320
596	Vita Harryman	1386 Old Boulevard	5407804067	703908907
603	Alicia Sharples	9627 Rustic Bluff Village	7034761501	276710180
611	Robbi Fishburn	3881 Colonial Wagon Hollow	8043611840	757425433
617	Teddy Ahlstrom	2562 Hazy Quail Concession	7035118224	434179191
625	Khalilah Rousseau	5872 Gentle Willow Green	5714288360	804469482
630	Tanna Sokolowski	2670 Noble Leaf Dell	5404808608	757806513
635	Sol Mcconville	9971 Lost Dale	4348924922	434635058

- Auth cookie creation (in login.aspx / auth code) – if cookie is created without HttpOnly/Secure flags.
- Upload vector (upload.aspx → wwwroot\uploads) – attacker can host JS (getcookies.htm / hair.svg) that runs in victim's browser.
- XSS/vector rendering (addalbum.aspx / listalbums.aspx) – stored XSS also allows JS to run in admin's browser.

Session Hijacking: Fixes



1. Set cookie flags when creating authentication cookie.
2. Regenerate session ID on login and on privilege changes
3. Enforce TLS/HSTS across site; set short timeout for session; re-authenticate for critical actions.
4. Removes JS-executable files from uploads and fix XSS vectors to prevent theft

Key Findings

Key Findings:

- Multiple critical web vulnerabilities identified: XSS, CSRF, Session Hijacking, Unrestricted File Upload.
- Exploits demonstrated real-world impact: account takeover, cookie theft, privilege escalation, and potential code execution.
- Root causes include: weak session management, lack of input/output validation, permissive file uploads, and missing anti-CSRF protections.
- Business & operational risks: data breaches, regulatory violations, loss of user trust, and potential financial losses.

Takeaways

Takeaways

- Always validate and encode all user input/output.
- Enforce strong session management (HttpOnly, Secure, SameSite cookies).
- Restrict and sanitize file uploads, and disable execution in upload directories.
- Implement anti-CSRF tokens and convert sensitive GET requests to POST.
- Monitor and alert on suspicious activity (unexpected uploads, sudden privilege changes, or external requests).
- Test in isolated environments before deploying changes to production.