ZAP Scanning Report

Generated with ZAP on Sat 12 Aug 2023, at 16:32:31

ZAP Version: 2.13.0

Contents

About this report

Report parameters

Contexts

No contexts were selected, so all contexts were included by default.

Sites

The following sites were included:

- `https://google-gruyere.appspot.com`
- `http://google-gruyere.appspot.com`

(If no sites were selected, all sites were included by default.)

An included site must also be within one of the included contexts for its data to be included in the report.

Risk levels

Included: `High`, `Medium`, `Low`, `Informational`

Excluded: None

Confidence levels

Included: `User Confirmed`, `High`, `Medium`, `Low`

Excluded: `User Confirmed`, `High`, `Medium`, `Low`, `False Positive`

Summaries

Alert counts by risk and confidence

This table shows the number of alerts for each level of risk and confidence included in the report.

(The percentages in brackets represent the count as a percentage of the total number of alerts included in the report, rounded to one decimal place.)

		Confidence				
		User Confirmed	High	Medium	Low	Total
Risk	High	0 (0.0%)	0 (0.0%)	1 (5.6%)	0 (0.0%)	1 (5.6%)
	Medium	0 (0.0%)	1 (5.6%)	1 (5.6%)	1 (5.6%)	3 (16.7%)
	Low	0 (0.0%)	1 (5.6%)	4 (22.2%)	0 (0.0%)	5 (27.8%)
	Informational	0 (0.0%)	1 (5.6%)	4 (22.2%)	4 (22.2%)	9 (50.0%)
	Total	0 (0.0%)	3 (16.7%)	10 (55.6%)	5 (27.8%)	18 (100%)

Alert counts by site and risk

This table shows, for each site for which one or more alerts were raised, the number of alerts raised at each risk level.

Alerts with a confidence level of "False Positive" have been excluded from these counts.

(The numbers in brackets are the number of alerts raised for the site at or above that risk level.)

		Risk			
		High (= High)	Medium (>= Medium)	Low (>= Low)	Informational (>= Informational)
Site	https://google-gruyere.appspot.com	0 (0)	0 (0)	1 (1)	1 (2)
	http://google-gruyere.appspot.com	1 (1)	1 (2)	3 (5)	5 (10)

Alert counts by alert type

This table shows the number of alerts of each alert type, together with the alert type's risk level.

(The percentages in brackets represent each count as a percentage, rounded to one decimal place, of the total number of alerts included in this report.)

Alert type	Risk	Count
Cross Site Scripting (Reflected)	High	4 (22.2%)
Absence of Anti-CSRF Tokens	Medium	11 (61.1%)
Content Security Policy (CSP) Header Not Set	Medium	87 (483.3%)
Missing Anti-clickjacking Header	Medium	85 (472.2%)
Big Redirect Detected (Potential Sensitive Information Leak)	Low	1 (5.6%)
Cookie No HttpOnly Flag	Low	5
Total		18

Alert type	Risk	Count
		(27.8%)
Cookie without SameSite Attribute	Low	5 (27.8%)
Strict-Transport-Security Header Not Set	Low	28 (155.6%)
X-Content-Type-Options Header Missing	Low	107 (594.4%)
Charset Mismatch (Header Versus Meta Content-Type Charset)	Informational	12 (66.7%)
Cookie Poisoning	Informational	5 (27.8%)
GET for POST	Informational	1 (5.6%)
Information Disclosure - Suspicious Comments	Informational	2 (11.1%)
Modern Web Application	Informational	23 (127.8%)
Re-examine Cache-control Directives	Informational	17 (94.4%)
Retrieved from Cache	Informational	31 (172.2%)
Session Management Response Identified	Informational	8 (44.4%)
User Agent Fuzzer	Informational	144 (800.0%)
Total		18

Alerts

Risk=High, Confidence=Medium (1)

http://google-gruyere.appspot.com (1)

Cross Site Scripting (Reflected) (1)

▼ POST http://google-gruyere.appspot.com/366974477833105008145397588284477084980/upload2

Alert tags

- OWASP_2021_A03
- WSTG-v42-INPV-01
- OWASP_2017_A07

Alert description

Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology.

When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise.

There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based.

Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash.

Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.

Request

▼ Request line and header section (702 bytes)

```
POST http://google-
gruyere.appspot.com/36697447783310500
8145397588284477084980/upload2
HTTP/1.1
host: google-gruyere.appspot.com
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,appli
cation/xml;q=0.9,image/avif,image/web
p,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://google-
gruyere.appspot.com/36697447783310500
8145397588284477084980/upload.gtl
Content-Type: multipart/form-data;
boundary=---------------------------
-3808664488903297968366864160
content-length: 264
Origin: http://google-
gruyere.appspot.com
Connection: keep-alive
Cookie:
GRUYERE=86969984|SOPJamLP||author
Upgrade-Insecure-Requests: 1
```

▼ Request body (264 bytes)

```
----------------------------
-3808664488903297968366864160
Content-Disposition: form-data;
```

```
name="upload_file"; filename="</div>
<scrIpt>alert(1);</scRipt><div>
Content-Type: application/octet-
stream


-----------------------------
-3808664488903297968366864160--
```

Response

▼ Status line and header section (249 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-type: text/html
Pragma: no-cache
X-XSS-Protection: 0
X-Cloud-Trace-Context:
c29925e57ed3b1b39324fedc996ee385
Date: Sat, 12 Aug 2023 17:31:19 GMT
Server: Google Frontend
Content-Length: 2814
```

▶ Response body (2814 bytes)

Parameter `upload_file`

Attack `</div><scrIpt>alert(1);</scRipt><div>`

Evidence `</div><scrIpt>alert(1);</scRipt><div>`

Solution

Phase: Architecture and Design

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-

XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

Phases: Implementation; Architecture and Design

Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies.

For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters.

Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed.

Phase: Architecture and Design

For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.

If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.

Phase: Implementation

For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping.

To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set.

Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list). However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue."

Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

Risk=`Medium`, Confidence=`High` (1)

Risk=`Medium`, Confidence=`Medium` (1)

Risk=`Medium`, Confidence=`Low` (1)

http://google-gruyere.appspot.com (1)

Absence of Anti-CSRF Tokens (1)

▼ GET http://google-gruyere.appspot.com/366974477833105008145397588284477084980/newaccount.gtl

Alert tags

- OWASP_2021_A01
- WSTG-v42-SESS-05
- OWASP_2017_A05

Alert description

No Anti-CSRF tokens were found in a HTML submission form.

A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

CSRF attacks are effective in a number of situations, including:

* The victim has an active session on the target site.

* The victim is authenticated via HTTP auth on the target site.

* The victim is on the same local network as the target site.

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

Other info

No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "action" "is_author" "pw" "uid"].

Request

▼ Request line and header section (392 bytes)

```
GET http://google-gruyere.appspot.com/366974477833105008145397588284477084980/newaccount.gtl HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36
```

```
pragma: no-cache
cache-control: no-cache
referer: http://google-
gruyere.appspot.com/36697447783310500
814539758828447708498 0/
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (253 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-type: text/html
Pragma: no-cache
X-XSS-Protection: 0
X-Cloud-Trace-Context:
2e4430650443e9b7b949ce9657ca799c;o=1
Date: Sat, 12 Aug 2023 17:24:04 GMT
Server: Google Frontend
Content-Length: 2961
```

▶ Response body (2961 bytes)

Evidence

```
<form method='get'
action='/36697447783310500814539758828
4477084980/saveprofile'>
```

Solution

Phase: Architecture and Design

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Phase: Implementation

Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.

Phase: Architecture and Design

Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).

Note that this can be bypassed using XSS.

Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.

Note that this can be bypassed using XSS.

Use the ESAPI Session Management control.

This control includes a component for CSRF.

Do not use the GET method for any request that triggers a state change.

Phase: Implementation

Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

Risk=Low, Confidence=High (1)

https://google-gruyere.appspot.com (1)

Strict-Transport-Security Header Not Set (1)

▼ GET https://google-gruyere.appspot.com/start.

Alert tags

- OWASP_2021_A05
- OWASP_2017_A06

Alert description

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTPS connections (i.e. HTTP layered over TLS/SSL). HSTS is an IETF standards track protocol and is specified in RFC 6797.

Request

▼ Request line and header section (370 bytes)

```
GET https://google-gruyere.appspot.com/start. HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: http://google-gruyere.appspot.com/part1
Cookie: GRUYERE_ID=640565808097385913255125469665165568921
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (263 bytes)

```
HTTP/1.1 404 Not Found
Content-Type: text/plain;
charset=UTF-8
X-Cloud-Trace-Context:
01be548e4044fb51aa11591e35f16d46
Date: Sat, 12 Aug 2023 17:24:09 GMT
Server: Google Frontend
Content-Length: 52
Alt-Svc: h3=":443"; ma=2592000,h3-
29=":443"; ma=2592000
```

▼ Response body (52 bytes)

```
404 Not Found

The resource could not be found.
```

Solution

Ensure that your web server, application server, load balancer, etc. is configured to enforce Strict-Transport-Security.

Risk=Low, Confidence=Medium (4)

http://google-gruyere.appspot.com (3)

Big Redirect Detected (Potential Sensitive Information Leak) (1)

▼ `GET http://google-gruyere.appspot.com/366974477833105008145397588284477084980/newsnippet2?snippet`

Alert tags

- [OWASP_2021_A04]
- [WSTG-v42-INFO-05]
- [OWASP_2017_A03]

Alert description

The server has responded with a redirect that seems to provide a large response. This may indicate that although the server sent a redirect it also responded with body content (which may include sensitive details, PII, etc.).

Other info

Location header URI length: 86 [http://google-gruyere.appspot.com/366974477833105008145397588284477084980/snippets.gtl].

Predicted response size: 386.

Response Body Length: 2,436.

Request

▼ Request line and header section (481 bytes)

```
GET http://google-
gruyere.appspot.com/3669744778331050
08145397588284477084980/newsnippet2?
snippet HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT
10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/114.0.0.0
Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: http://google-
gruyere.appspot.com/3669744778331050
```

```
08145397588284477084980/newsnippet.g
tl
Cookie: GRUYERE=;
GRUYERE_ID=640565808097385913255125469665165568921
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (350 bytes)

```
HTTP/1.1 302 Found
Cache-Control: no-cache
Location: http://google-
gruyere.appspot.com/366974477833105008145397588284477084980/snippets.gtl
Pragma: no-cache
Content-type: text/html
X-XSS-Protection: 0
X-Cloud-Trace-Context:
d2ef3940c947f7ed0baaa9e805b78207
Date: Sat, 12 Aug 2023 17:24:11 GMT
Server: Google Frontend
Content-Length: 2436
```

▶ Response body (2436 bytes)

Solution

Ensure that no sensitive information is leaked via redirect responses. Redirect responses should have almost no content.

Cookie No HttpOnly Flag (1)

▼ GET http://google-gruyere.appspot.com/start

Alert tags

- OWASP_2021_A05
- WSTG-v42-SESS-02

- OWASP_2017_A06

Alert description

A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.

Request

▼ Request line and header section (313 bytes)

```
GET http://google-
gruyere.appspot.com/start HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT
10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/114.0.0.0
Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: http://google-
gruyere.appspot.com/robots.txt
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (354 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: text/html;
charset=utf-8
Pragma: no-cache
Set-Cookie:
GRUYERE_ID=6405658080973859132551254
69665165568921; Path=/
X-Cloud-Trace-Context:
```

```
dc5aa62e80b3034ec74e77874c7b2861
Date: Sat, 12 Aug 2023 17:24:05 GMT
Server: Google Frontend
Content-Length: 681
Expires: Sat, 12 Aug 2023 17:24:05
GMT
```

▼ Response body (681 bytes)

```
    <HTML>

  <STYLE>
  body, th, td, form {
    font-family: Verdana, Arial,
Helvetica, sans-serif;
    font-size: 12px;
  }
  h1 { color: #dd0000; }
  </STYLE>
    <TITLE>Start Gruyere</TITLE>
    <BODY>
    <H1>Start Gruyere</H1>
    Your Gruyere instance id is
640565808097385913255125469665165568
921.

    <br><br><span style="color:red">
<b>WARNING: Gruyere is not secure.
<br>
    Do not upload any personal or
private data.</b></span>

    <br><br>By using Gruyere you
agree to the
    <A
href="https://www.google.com/intl/en
/policies/terms/">terms of
service</A>.

    <H2><A
HREF="/64056580809738591325512546966
```

```
5165568921">Agree &amp; Start</A>
</H2>
    </BODY></HTML>
```

Parameter

```
GRUYERE_ID
```

Evidence

```
Set-Cookie: GRUYERE_ID
```

Solution

Ensure that the HttpOnly flag is set for all cookies.

Cookie without SameSite Attribute (1)

▼ `GET http://google-gruyere.appspot.com/start`

Alert tags

- OWASP_2021_A01
- WSTG-v42-SESS-02
- OWASP_2017_A05

Alert description

A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.

Request

▼ Request line and header section (313 bytes)

```
GET http://google-
gruyere.appspot.com/start HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT
10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/114.0.0.0
Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: http://google-
```

```
gruyere.appspot.com/robots.txt
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (354 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: text/html;
charset=utf-8
Pragma: no-cache
Set-Cookie:
GRUYERE_ID=6405658080973859132551254
69665165568921; Path=/
X-Cloud-Trace-Context:
dc5aa62e80b3034ec74e77874c7b2861
Date: Sat, 12 Aug 2023 17:24:05 GMT
Server: Google Frontend
Content-Length: 681
Expires: Sat, 12 Aug 2023 17:24:05
GMT
```

▼ Response body (681 bytes)

```
    <HTML>

  <STYLE>
  body, th, td, form {
    font-family: Verdana, Arial,
Helvetica, sans-serif;
    font-size: 12px;
  }
  h1 { color: #dd0000; }
  </STYLE>
    <TITLE>Start Gruyere</TITLE>
    <BODY>
    <H1>Start Gruyere</H1>
    Your Gruyere instance id is
```

```
640565808097385913255125469665165568
921.

    <br><br><span style="color:red">
<b>WARNING: Gruyere is not secure.
<br>
    Do not upload any personal or
private data.</b></span>

    <br><br>By using Gruyere you
agree to the
    <A
href="https://www.google.com/intl/en
/policies/terms/">terms of
service</A>.

    <H2><A
HREF="/6405658080973859132551254696651655689 21">Agree &amp; Start</A>
</H2>
    </BODY></HTML>
```

Parameter	`GRUYERE_ID`
Evidence	`Set-Cookie: GRUYERE_ID`
Solution	Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.

Risk=Informational, Confidence=High (1)

http://google-gruyere.appspot.com (1)

GET for POST (1)

▼ GET http://google-gruyere.appspot.com/366974477833105008145397588284477084980/upload2

Alert tags

- OWASP_2021_A04
- WSTG-v42-CONF-06
- OWASP_2017_A06

Alert description

A request that was originally observed as a POST was also accepted as a GET. This issue does not represent a security weakness unto itself, however, it may facilitate simplification of other attacks. For example if the original POST is subject to Cross-Site Scripting (XSS), then this finding may indicate that a simplified (GET based) XSS may also be possible.

Request

▼ Request line and header section (540 bytes)

```
GET http://google-
gruyere.appspot.com/3669744778331050
0814539758828447 7084980/upload2?
upload_file=test_file.txt HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT
10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/114.0.0.0
Safari/537.36
pragma: no-cache
cache-control: no-cache
content-type: application/x-www-
form-urlencoded
referer: http://google-
gruyere.appspot.com/3669744778331050
08145397588284477084980/upload.gtl
Cookie: GRUYERE=;
GRUYERE_ID=6405658080973859132551254
69665165568921
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (242 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: text/html;
charset=utf-8
Pragma: no-cache
X-Cloud-Trace-Context:
457208c18bead2117d6ddbaa35b88e44
Date: Sat, 12 Aug 2023 18:45:59 GMT
Server: Google Frontend
Content-Length: 224
```

▼ Response body (224 bytes)

```
        <HTML><BODY>
        <H1
style="color:red">Gruyere System
Alert</H1>
        <TT style="font-size: 150%">
        Exception: 'unicode' object
has no attribute 'filename'
        </TT>

        </BODY></HTML>
```

Evidence

```
GET http://google-
gruyere.appspot.com/3669744778331050
08145397588284477084980/upload2?
upload_file=test_file.txt HTTP/1.1
```

Solution

Ensure that only POST is accepted where POST is expected.

Risk=`Informational`, Confidence=`Medium` (4)

`http://google-gruyere.appspot.com` (2)

Session Management Response Identified (1)

▼ GET http://google-gruyere.appspot.com/start

Alert tags

Alert description

The given response has been identified as containing a session management token. The 'Other Info' field contains a set of header tokens that can be used in the Header Based Session Management Method. If the request is in a context which has a Session Management Method set to "Auto-Detect" then this rule will change the session management to use the tokens identified.

Other info

cookie:GRUYERE_ID

Request

▼ Request line and header section (313 bytes)

```
GET http://google-gruyere.appspot.com/start HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: http://google-gruyere.appspot.com/robots.txt
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (354 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: text/html;
charset=utf-8
Pragma: no-cache
Set-Cookie:
GRUYERE_ID=6405658080973859132551254
69665165568921; Path=/
X-Cloud-Trace-Context:
dc5aa62e80b3034ec74e77874c7b2861
Date: Sat, 12 Aug 2023 17:24:05 GMT
Server: Google Frontend
Content-Length: 681
Expires: Sat, 12 Aug 2023 17:24:05
GMT
```

▼ Response body (681 bytes)

```
    <HTML>

  <STYLE>
  body, th, td, form {
    font-family: Verdana, Arial,
Helvetica, sans-serif;
    font-size: 12px;
  }
  h1 { color: #dd0000; }
  </STYLE>
    <TITLE>Start Gruyere</TITLE>
    <BODY>
    <H1>Start Gruyere</H1>
    Your Gruyere instance id is
640565808097385913255125469665165568
921.

    <br><br><span style="color:red">
<b>WARNING: Gruyere is not secure.
<br>
    Do not upload any personal or
private data.</b></span>
```

```
    <br><br>By using Gruyere you
agree to the
    <A
href="https://www.google.com/intl/en
/policies/terms/">terms of
service</A>.

    <H2><A
HREF="/64056580809738591325512546966
5165568921">Agree &amp; Start</A>
</H2>
    </BODY></HTML>
```

Parameter GRUYERE_ID

Evidence 640565808097385913255125469665165568921

Solution This is an informational alert rather than a vulnerability and so there is nothing to fix.

User Agent Fuzzer (1)

▼ GET http://google-gruyere.appspot.com/366974477833105008145397588284477084980/

Alert tags

Alert description Check for differences in response based on fuzzed User Agent (eg. mobile sites, access as a Search Engine Crawler). Compares the response statuscode and the hashcode of the response body with the original response.

Request ▼ Request line and header section (371 bytes)

```
GET http://google-
gruyere.appspot.com/3669744778331050
```

```
08145397588284477084980/ HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (249 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-type: text/html
Pragma: no-cache
X-XSS-Protection: 0
X-Cloud-Trace-Context: bd72ab37384751e5e20777a8e3e6565f
Date: Sat, 12 Aug 2023 18:46:00 GMT
Server: Google Frontend
Content-Length: 4286
```

▶ Response body (4286 bytes)

Parameter	Header User-Agent
Attack	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)

Risk=Informational, Confidence=Low (4)

https://google-gruyere.appspot.com (1)

Re-examine Cache-control Directives (1)

▼ GET https://google-gruyere.appspot.com/code/

Alert tags

- WSTG-v42-ATHN-06

Alert description

The cache-control header has not been set properly or is missing, allowing the browser and proxies to cache content. For static assets like css, js, or image files this might be intended, however, the resources should be reviewed to ensure that no sensitive content will be cached.

Request

▼ Request line and header section (369 bytes)

```
GET https://google-
gruyere.appspot.com/code/ HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT
10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/114.0.0.0
Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: http://google-
gruyere.appspot.com/part1
Cookie:
GRUYERE_ID=6405658080973859132551254
69665165568921
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (299 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: text/html;
charset=utf-8
X-Cloud-Trace-Context:
642634257df794a51538181e72891e85
Date: Sat, 12 Aug 2023 17:24:09 GMT
Server: Google Frontend
Content-Length: 354
Alt-Svc: h3=":443"; ma=2592000,h3-
29=":443"; ma=2592000
```

▼ Response body (354 bytes)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Frameset//EN"

"https://www.w3.org/TR/html4/framese
t.dtd">
<HTML>
<HEAD>
<TITLE>Gruyere Code</TITLE>
</HEAD>
<FRAMESET cols="165,100%">
  <FRAME
src="/static/codeindex.html">
  <FRAME name="codepage" src="">
  <NOFRAMES>
    <A
href="/static/codeindex/html">Code
Index</A>
  </NOFRAMES>
</FRAMESET>
</HTML>
```

Parameter `cache-control`

Evidence `no-cache`

Solution For secure content, ensure the cache-control HTTP header is set with "no-cache, no-store, must-revalidate". If an asset should be cached consider setting the directives "public, max-age, immutable".

http://google-gruyere.appspot.com (2)

Charset Mismatch (Header Versus Meta Content-Type Charset) (1)

▼ GET http://google-gruyere.appspot.com/

Alert tags	
Alert description	This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.
Other info	There was a charset mismatch between the HTTP Header and the META content-type encoding declarations: [utf-8] and [ISO-8859-1] do not match.

Request

▼ Request line and header section (299 bytes)

```
GET http://google-gruyere.appspot.com/ HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: http://google-gruyere.appspot.com/1
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (244 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: text/html; charset=utf-8
X-Cloud-Trace-Context: 97fd933765e74046e841be084c6dc57c
Date: Sat, 12 Aug 2023 17:24:05 GMT
Server: Google Frontend
Content-Length: 11506
```

▶ Response body (11506 bytes)

Solution

Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Cookie Poisoning (1)

▼ `GET http://google-gruyere.appspot.com/366974477833105008145397588284477084980/login?pw=ZAP&uid=ZAP`

Alert tags

- OWASP_2021_A03
- OWASP_2017_A01

Alert description

This check looks at user-supplied input in query string parameters and POST data to identify where cookie parameters might be controlled. This is called a cookie poisoning attack, and becomes exploitable when an attacker can manipulate the cookie in various ways. In some cases this will not be exploitable, however, allowing URL parameters to set cookie values is generally considered a bug.

Other info

An attacker may be able to poison cookie values through URL parameters. Try injecting a semicolon to see if you can add cookie values (e.g. name=controlledValue;name=anotherValue;).

This was identified at:

http://google-gruyere.appspot.com/366974477833105008145397588284477084980/login?pw=ZAP&uid=ZAP

User-input was found in the following cookie:

GRUYERE=26087470|ZAP||author; path=/366974477833105008145397588284477084980

The user input was:

pw=ZAP

Request

▼ Request line and header section (403 bytes)

```
GET http://google-
gruyere.appspot.com/3669744778331050
08145397588284477084980/login?
pw=ZAP&uid=ZAP HTTP/1.1
host: google-gruyere.appspot.com
user-agent: Mozilla/5.0 (Windows NT
10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/114.0.0.0
Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: http://google-
gruyere.appspot.com/3669744778331050
08145397588284477084980/login
```

▼ Request body (0 bytes)

Response

▼ Status line and header section (378 bytes)

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-type: text/html
Pragma: no-cache
Set-Cookie:
GRUYERE=26087470|ZAP||author;
path=/366974477833105008145397588284
477084980
X-XSS-Protection: 0
X-Cloud-Trace-Context:
6fb59902487646dd3e268b88011e19f0
Date: Sat, 12 Aug 2023 17:24:06 GMT
Server: Google Frontend
Content-Length: 4211
Expires: Sat, 12 Aug 2023 17:24:06
```

GMT

▶ Response body (4211 bytes)

Parameter	pw
Solution	Do not allow user input to control cookie names and values. If some query string parameters must be set in cookie values, be sure to filter out semicolon's that can serve as name/value pair delimiters.

Appendix

Alert types

This section contains additional information on the types of alerts in the report.

Cross Site Scripting (Reflected)

Source	raised by an active scanner (Cross Site Scripting (Reflected))
CWE ID	79
WASC ID	8
Reference	▪ http://projects.webappsec.org/Cross-Site-Scripting ▪ http://cwe.mitre.org/data/definitions/79.html

Absence of Anti-CSRF Tokens

Source	raised by a passive scanner (Absence of Anti-CSRF Tokens)
CWE ID	352
WASC ID	9
Reference	▪ http://projects.webappsec.org/Cross-Site-Request-Forgery ▪ http://cwe.mitre.org/data/definitions/352.html

Content Security Policy (CSP) Header Not Set

Source	raised by a passive scanner (Content Security Policy (CSP) Header Not Set)
CWE ID	693
WASC ID	15
Reference	▪ https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy ▪ https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html ▪ http://www.w3.org/TR/CSP/ ▪ http://w3c.github.io/webappsec/specs/content-security-policy/csp-specification.dev.html ▪ http://www.html5rocks.com/en/tutorials/security/content-security-policy/

- http://caniuse.com/#feat=contentsecuritypolicy
- http://content-security-policy.com/

Missing Anti-clickjacking Header

Source	raised by a passive scanner (Anti-clickjacking Header)
CWE ID	1021
WASC ID	15
Reference	▪ https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

Big Redirect Detected (Potential Sensitive Information Leak)

Source	raised by a passive scanner (Big Redirect Detected (Potential Sensitive Information Leak))
CWE ID	201
WASC ID	13

Cookie No HttpOnly Flag

Source	raised by a passive scanner (Cookie No HttpOnly Flag)
CWE ID	1004
WASC ID	13
Reference	▪ https://owasp.org/www-community/HttpOnly

Cookie without SameSite Attribute

Source	raised by a passive scanner (Cookie without SameSite Attribute)
CWE ID	1275
WASC ID	13
Reference	▪ https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site

Strict-Transport-Security Header Not Set

Source	raised by a passive scanner (Strict-Transport-Security Header)
CWE ID	319
WASC ID	15
Reference	▪ https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html ▪ https://owasp.org/www-community/Security_Headers ▪ http://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security ▪ http://caniuse.com/stricttransportsecurity ▪ http://tools.ietf.org/html/rfc6797

X-Content-Type-Options Header Missing

Source	raised by a passive scanner (X-Content-Type-Options Header Missing)

CWE ID	693
WASC ID	15
Reference	▪ http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx ▪ https://owasp.org/www-community/Security_Headers

Charset Mismatch (Header Versus Meta Content-Type Charset)

Source	raised by a passive scanner (Charset Mismatch)
CWE ID	436
WASC ID	15
Reference	▪ http://code.google.com/p/browsersec/wiki/Part2#Character_set_handling_and_detection

Cookie Poisoning

Source	raised by a passive scanner (Cookie Poisoning)
CWE ID	20
WASC ID	20
Reference	▪ http://websecuritytool.codeplex.com/wikipage?title=Checks#user-controlled-cookie

GET for POST

Source	raised by an active scanner (GET for POST)
CWE ID	16

WASC ID	20

Information Disclosure - Suspicious Comments

Source	raised by a passive scanner (Information Disclosure - Suspicious Comments)
CWE ID	200
WASC ID	13

Modern Web Application

Source	raised by a passive scanner (Modern Web Application)

Re-examine Cache-control Directives

Source	raised by a passive scanner (Re-examine Cache-control Directives)
CWE ID	525
WASC ID	13
Reference	▪ https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#web-content-caching ▪ https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control ▪ https://grayduck.mn/2021/09/13/cache-control-recommendations/

Retrieved from Cache

Source	raised by a passive scanner (Retrieved from Cache)
Reference	- https://tools.ietf.org/html/rfc7234 - https://tools.ietf.org/html/rfc7231 - http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html (obsoleted by rfc7234)

Session Management Response Identified

Source	raised by a passive scanner (Session Management Response Identified)
Reference	- https://www.zaproxy.org/docs/desktop/addons/authentication-helper/session-mgmt-id

User Agent Fuzzer

Source	raised by an active scanner (User Agent Fuzzer)
Reference	- https://owasp.org/wstg