Java Eclipse Jetty Report: Incorrect Parsing Priority of the IPv6 Hostname Delimeter

0x01 Affected components

1.1 Affected components

- **Java Eclipse Jetty**
- **Versions:** tested in 12.0.10
- **CLAIMS TO FOLLOW: RFC-3986**

1.2 Attack scenario

The threat model illustrated in Figure 1 explains the security risks in web systems caused by inconsistent URL parsing. Attackers initiate requests to web systems by constructing ambiguous URLs. These requests first go through a preprocessor for security checks. Preprocessors typically handle tasks such as permission verification, URL whitelist and blacklist checks, and URL normalization to ensure that the requested access is to authorized resources. However, due to the possibility of preprocessors and executors (such as browsers, API routers, requesters, etc.) using different programming languages or following different specification standards for URL parsing, the same URL string may be parsed into different target resource locations. Specifically, as shown in the figure, the preprocessor may parse the URL into a legitimate resource location A and pass security verification, while the executor parses the same URL into another sensitive resource location B and ultimately executes the actual request targeting sensitive resource B. This parsing discrepancy allows attackers to cleverly bypass the preprocessor's security mechanisms, gaining access to sensitive or unauthorized resources, thus posing potential security threats.

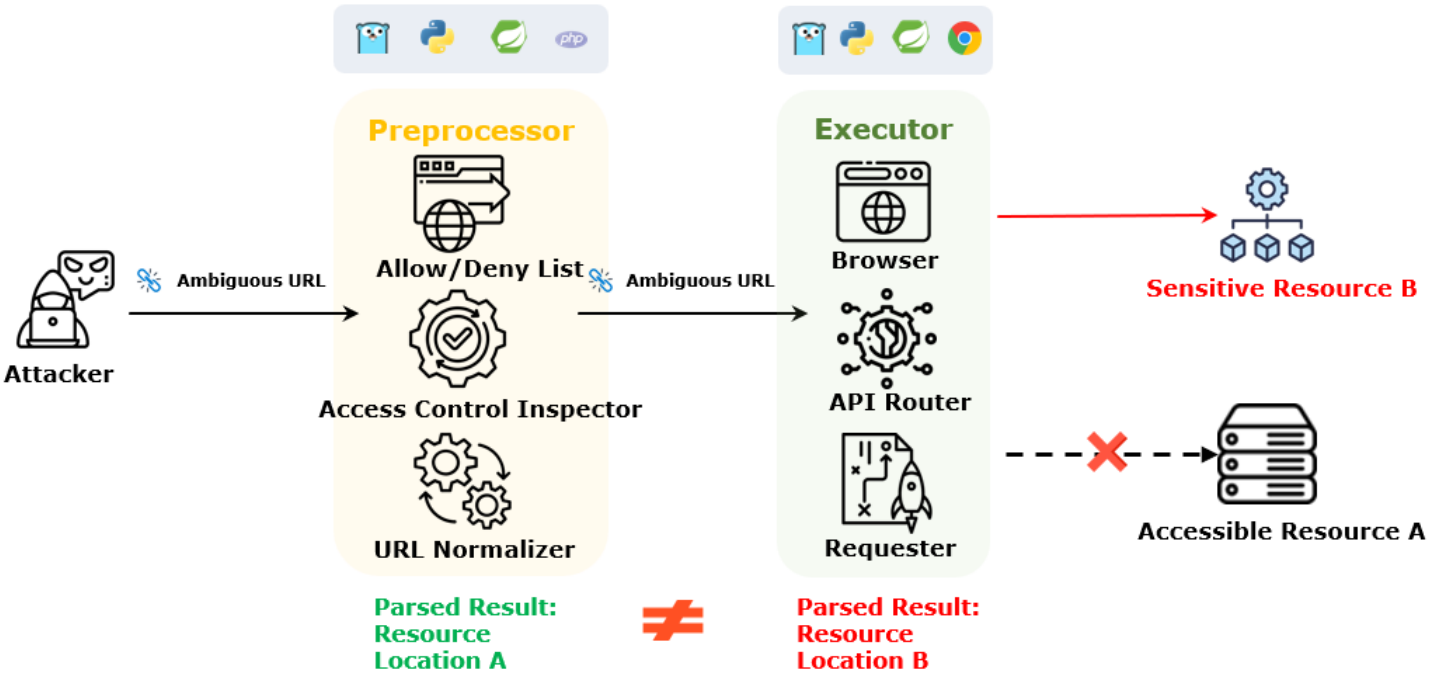

Figure 1: Principle of URL semantic gap attack caused by inconsistent URL parsing.

Through our research, we have found that for this attack scenario, URL semantic gap attack may lead to three main types of security vulnerabilities:

1. Server-side request forgery: The attacker bypasses the domain name check of the preprocessor, causing the executor to make requests to the internal network or unauthorized external servers.
2. Open redirection: The attacker bypasses the domain name whitelist check of the redirection URL, redirecting the browser request to a malicious website.
3. Access control bypass: The attacker bypasses access control policies based on HTTP request paths to gain unauthorized access to protected resources.

0x02 Affected components

2.1 Overview of the Issue

The following case shows the parsing result of deformed URL `http://[normal.com@]vulndetector.com/`. According to relevant URL standard, it should be marked as an invalid URL.

payload	jetty	urllib3(python)	furl(python)
http://[normal.com@]vulndetector.com/	hostname=[normal.com@]	Invalid URL	Invalid URL

We notice that currently several parsers will parse such URL normally, and the result of them are not the same, which may cause SSRF Bypass.

payload	jetty	SpringFramework	chromium
http://normal.com [user@vulndetector].com/	[normal.com@vulndetector]	normal.com	Invalid URL
http://normal.com[@]vulndetector.com/	normal.com[@]	normal.com	Invalid URL

2.2 Definition of this parsing behavior in international standards

RFC parsing standard

According to RFC 3986, the general structure of a URL is as follows:

Code block

```
URI = scheme ":" hier-part [ "?" query ] [ "#" fragment ]
hier-part = "//" authority path-abempty
authority = [ userinfo "@" ] host [ ":" port ]
host = IP-literal / IPv4address / reg-name
userinfo    = *( unreserved / pct-encoded / sub-delims / ":" )
unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"
pct-encoded = "%" HEXDIG HEXDIG
sub-delims  = "!" / "$" / "&" / "'" / "(" / ")" / "*" / "+" / "," / ";" / "="
```

The `authority` section allows `userinfo@host` formats, but `userinfo` must not contain the `[` symbol, otherwise it will cause parsing errors.

In `http://normal.com [@vulndetector.com/` this URL:

- `normal.com [@vulndetector.com/` violates `authority` resolution rules because `normal.com [` is not a valid `userinfo` .
- RFC 3986 The parser should reject the URL with an error because `userinfo` cannot contain the character `[` .

WHATWG URL Parsing Standard

If there is an `[` or `]` instead of a pair of [] symbols, IPv6-unclosed validation error will be reported.

Code block

```
The host parser takes a scalar value string input with an optional boolean
isOpaque (default false), and then runs these steps. They return failure or a
host.
If input starts with U+005B ([), then:
If input does not end with U+005D (]), IPv6-unclosed validation error, return
failure.
```

```
4   Return the result of IPv6 parsing input with its leading U+005B ([) and
    trailing U+005D (]) removed.
```

In addition, the parsing of authority is also not allowed to appear [or] symbol, if it contains `host-invalid-code-point` should trigger an error

Code block

```
1   An opaque host (in a URL that is not special) contains a forbidden host code
    point.
2   Example: foo://exa[mple.org
```

Forbidden host code points include the following characters

Code block

```
1   A forbidden host code point is:
2   U+0000 NULL, U+0009 TAB, U+000A LF, U+000D CR, U+0020 SPACE, U+0023 (#),
3   U+002F (/), U+003A (:), U+003C (<), U+003E (>), U+003F (?), U+0040 (@),
4   U+005B ([), U+005C (\), U+005D (]), U+005E (^), or U+007C (|).
```

Conclusion

Standard	`Authority` allows `[` ?	Analytic behavior
RFC 3986	No (unless used for IPv6)	Parsing failed
WHATWG	No (unless used for IPv6)	`invalid-credentials` + `host-invalid-code-point`

2.3 Security threat scenario - SSRF attack caused by inconsistent URL host resolution

1. **Attacker constructs URL** : `http://normal.com [@vulndetector.com/`

- In some applications (such as API gateways, firewalls , Cloud as a Service), URLs may be **securely validated by the resolver** before being handled by the **request sending resolver** .

2. **Security authentication parser** :
    - Resolve `host = normal.com` and assume the URL is safe.
    - Allow the request to pass without blocking it.

3. **Request send parser** :
    - Because of the `@` , the resolver resolves the `host` to a `vulndetector.com` and sends the request to the **server controlled by the attacker** .

Eventually the **server mistakenly thought the target was `normal.com` , but the actual access was `vulndetector.com` , and the attack was successful.**

2.4 Mitigation measures

Strictly follow the parsing rules of RFC 3986 and WHATWG URLs, and reject illegal `authority` formats.

0x03 Acknowledgments

Enze Wang @IPASSLAB && Tsinghua University

Jingcheng Yang @Tsinghua University

Zehui Miao @Tsinghua University