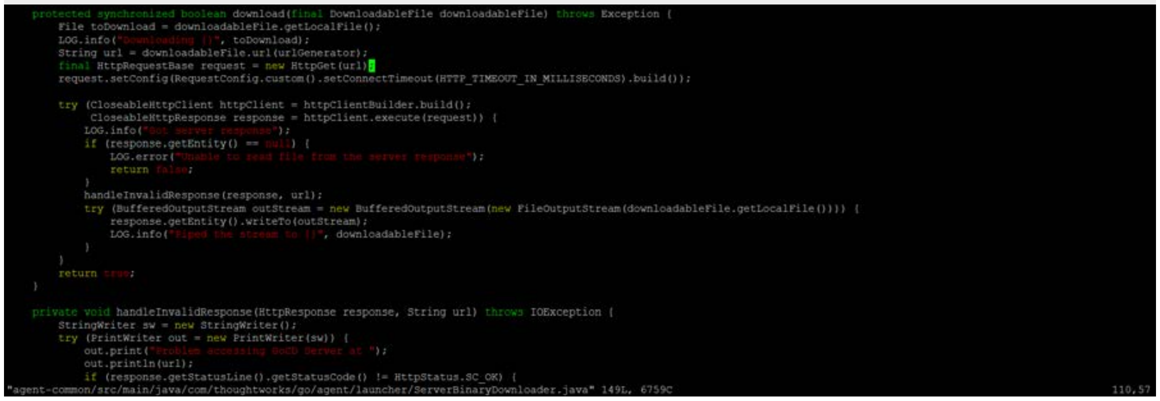
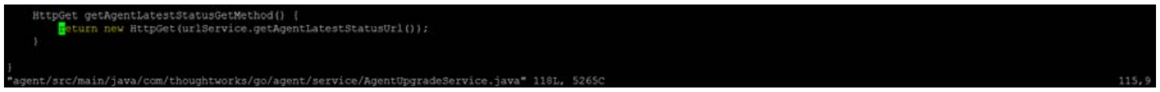
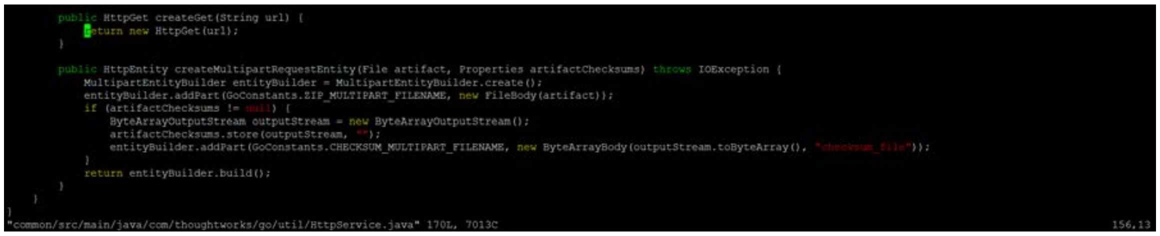
com/thoughtworks/go/agent/launcher/ServerBinaryDownloader.java	110	2	SECURITY	HTTP_PARAMETER_POLLUTION	ℹ	💡
com/thoughtworks/go/agent/service/AgentUpgradeService.java	115	2	SECURITY	HTTP_PARAMETER_POLLUTION	ℹ	💡
com/thoughtworks/go/util/HttpService.java	156	2	SECURITY	HTTP_PARAMETER_POLLUTION	ℹ	💡



```
    protected synchronized boolean download(final DownloadableFile downloadableFile) throws Exception {
        File toDownload = downloadableFile.getLocalFile();
        LOG.info("Downloading {}", toDownload);
        String url = downloadableFile.url(urlGenerator);
        final HttpRequestBase request = new HttpGet(url);
        request.setConfig(RequestConfig.custom().setConnectTimeout(HTTP_TIMEOUT_IN_MILLISECONDS).build());

        try (CloseableHttpClient httpClient = httpClientBuilder.build();
             CloseableHttpResponse response = httpClient.execute(request)) {
            LOG.info("Got server response");
            if (response.getEntity() == null) {
                LOG.error("Unable to read file from the server response");
                return false;
            }
            handleInvalidResponse(response, url);
            try (BufferedOutputStream outStream = new BufferedOutputStream(new FileOutputStream(downloadableFile.getLocalFile()))) {
                response.getEntity().writeTo(outStream);
                LOG.info("Piped the stream to {}", downloadableFile);
            }
        }
        return true;
    }

    private void handleInvalidResponse(HttpResponse response, String url) throws IOException {
        StringWriter sw = new StringWriter();
        try (PrintWriter out = new PrintWriter(sw)) {
            out.print("Problem accessing GoCD Server at ");
            out.println(url);
            if (response.getStatusLine().getStatusCode() != HttpStatus.SC_OK) {
"agent-common/src/main/java/com/thoughtworks/go/agent/launcher/ServerBinaryDownloader.java" 149L, 6759C                110,57
```



```
    HttpGet getAgentLatestStatusGetMethod() {
        return new HttpGet(urlService.getAgentLatestStatusUrl());
    }

}
"agent/src/main/java/com/thoughtworks/go/agent/service/AgentUpgradeService.java" 118L, 5265C                115,9
```



```
        public HttpGet createGet(String url) {
            return new HttpGet(url);
        }

        public HttpEntity createMultipartRequestEntity(File artifact, Properties artifactChecksums) throws IOException {
            MultipartEntityBuilder entityBuilder = MultipartEntityBuilder.create();
            entityBuilder.addPart(GoConstants.ZIP_MULTIPART_FILENAME, new FileBody(artifact));
            if (artifactChecksums != null) {
                ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
                artifactChecksums.store(outputStream, "");
                entityBuilder.addPart(GoConstants.CHECKSUM_MULTIPART_FILENAME, new ByteArrayBody(outputStream.toByteArray(), "checksum_file"));
            }
            return entityBuilder.build();
        }
    }
}
"common/src/main/java/com/thoughtworks/go/util/HttpService.java" 170L, 7013C                156,13
```

Concatenating user-controlled input into a URL

Concatenating unvalidated user input into a URL can allow an attacker to override the value of a request parameter. Attacker may be able to override existing parameter values, inject a new parameter or exploit variables out of a direct reach. HTTP Parameter Pollution (HPP) attacks consist of injecting encoded query string delimiters into other existing parameters. If a web application does not properly sanitize the user input, a malicious user may compromise the logic of the application to perform either client-side or server-side attacks.

In the following example the programmer has not considered the possibility that an attacker could provide a parameter lang such as en&user_id=1, which would enable him to change the user_id at will.

Vulnerable Code:
String lang = request.getParameter("lang"); GetMethod get = new GetMethod("http://www.host.com"); get.setQueryString("lang=" + lang + "&user_id=" + user_id); get.execute();

Solution:
Sanitize user input before using it in HTTP parameters.

References
CAPEC-460: HTTP Parameter Pollution (HPP)