November 05, 2020

Certificate Validation Disabled in Black Duck API Wrapper

These days, security practitioners place a considerable amount of effort into securing their organization's applications with secure software development lifecycle (SDLC) programs. One piece of the puzzle in an organization's secure SDLC is typically a set of security tools (SAST, DAST, IAST, SCA…) running in one or more CI/CD pipelines. These tools are meant to help the organization keep an eye on the security posture of their applications. As an example, SCA tools like Black Duck watch the third-party components utilized by applications to ensure that the components do not contain any known vulnerabilities. But who watches the watchers?

Optiv has identified a security vulnerability in the Black Duck Hub REST API Python project that is listed in the PyPI repository under the name "blackduck." This issue has been assigned the CVE identifier CVE-2020-27589. When reviewing the API wrapper's source code, Optiv found that despite having a configuration option for disabling HTTPS certificate validation, three instances existed where certificate validation was always disabled. This allowed network attackers to intercept and modify sensitive data in transit from the API wrapper to the Black Duck server.

Vulnerability Details

The API wrapper makes HTTPS requests to the Black Duck server using the Python requests library. For individual requests made using this library, certificate validation can be enabled or disabled using the optional argument "verify." Most of the methods in the API wrapper use the configuration option "insecure" to set the value of the "verify" argument. This allows the user to specify if certificate validation should be enforced or not.

```
def get_projects(self, limit=100, parameters={}):
  headers = self.get_headers()
  if limit:
     parameters.update({'limit': limit})
  url = self._get_projects_url() + self._get_parameter_string(parameters)
  headers['Accept'] = 'application/vnd.blackducksoftware.project-detail-4+json'
  logger.debug(f"Retrieving projects using url {url}")
  response = requests.get(url, headers=headers, verify = not self.config['insecure'])
  jsondata = response.json()
  return jsondata
```

Figure 1: HubRestApi.py version 0.0.51 line 673-682

The "upload_scan" method is used to upload scan results to the Black Duck server. The following application source code shows that the scan results are uploaded to the server via the Python requests library. Notice in both cases, the "verify" argument is set to the literal "False" instead of using the configuration option "insecure" as is done throughout the application. This means that the requests made using this method will always have certificate validation disabled, even when the user configures the library to enforce certificate validation. This instance was present in the API wrapper from version 0.0.28 - 0.0.52.

```
def upload_scan(self, filename):
  url = self.get_apibase() + "/scan/data/?mode=replace"
  headers = self.get_headers()
  if filename.endswith('.json') or filename.endswith('.jsonld'):
     headers['Content-Type'] = 'application/ld+json'
```

The “download_project_scans” method is used to download scan results from the Black Duck server. Notice that the same issue exists where the “verify” argument is set to the value “False” to disable certificate validation in all cases. This instance was present in the API wrapper from version 0.0.25 - 0.0.52.

```
def download_project_scans(self, project_name,version_name, output_folder=None):
  version = self.get_project_version_by_name(project_name,version_name)
  codelocations = self.get_version_codelocations(version)
  import os
  if output_folder:
    if not os.path.exists(output_folder):
      os.makedirs(output_folder, 0o755, True)

  result = []

  for item in codelocations['items']:
    links = item['_meta']['links']
    matches = [x for x in links if x['rel'] == 'enclosure']
    for m in matches:
      url = m['href']
      filename = url.split('/')[6]
      if output_folder:
          pathname = os.path.join(output_folder, filename)
      else:
          if not os.path.exists(project_name):
            os.mkdir(project_name)
          pathname = os.path.join(project_name, filename)
      responce = requests.get(url, headers=self.get_headers(), stream=True, verify=False)
      with open(pathname, "wb") as f:
          for data in responce.iter_content():
              f.write(data)
      result.append({filename, pathname})
  return result
```

Figure 3: HubRestApi.py version 0.0.51 line 1297-1324

Exploitation - bd-offline-scanning-solution

One of the publicly available projects that used a vulnerable version of the API wrapper was the “bd-offline-scanning-solution” project. This project, currently maintained by Synopsys, allows Black Duck users in restricted environments to perform offline scans and to upload the results to the Black Duck server at a later time. The source code below shows that the vulnerable “upload_scan” method is used by the project.

```
manifest = json.load(open(args.manifest, 'r'))

for file in manifest['scan_files']:
[...]
        #
        # Upload the scan file
        #
        logging.debug(msg)
        logging.debug(hub.upload_scan(file_to_upload))
        if temp_file and not args.keep:
            logging.debug("removing temp file {}".format(file_to_upload))
            os.remove(file_to_upload)
        elif temp_file:
            logging.debug("preserving temp file {}".format(file_to_upload))
```

Figure 4: blackducksoftware/bd-offline-scanning-solution/upload_scans.py line 45-107

If an attacker has knowledge of this vulnerability and is able to intercept and modify network traffic between the system running the offline scanner solution and the Black Duck server, they could both view or modify scan results in transit to the Black Duck server. This would allow the attacker to know what components the scanned application is using so that they could research vulnerabilities in the components separately. Additionally, an advanced attacker could potentially modify the bill of materials in flight, causing some components that were identified by Black Duck at scan time to not show up in the Black Duck web interface. An advanced attacker could use this ability to intentionally filter out

```
response = requests.get(url, headers=headers, verify = not self.config['insecure'])
```

The vendor released version 0.0.53 which includes a fix for this issue. Additionally, Optiv recommends that users configure the API wrapper to enforce certificate validation at all times by setting the configuration value “insecure” to “false.”

Vulnerability Disclosure Timeline

- September 22, 2020 - Optiv attempted to contact maintainer
- September 23, 2020 - Optiv attempted to contact maintainer
- October 19, 2020 - Optiv contacted general security inbox
- October 19, 2020 - Optiv submitted support case to vendor
- October 19, 2020 - Vendor acknowledged the issue and agreed to release the fixed version
- October 20, 2020 - Optiv disclosed to MITRE Corporation within CVE request
- October 22, 2020 - Vendor released the fixed version on PyPI 0.0.53
- October 27, 2020 - Optiv met with vendor to discuss coordinated disclosure
- October 28, 2020 - CVE-2020-27589 assigned by CNA (MITRE Corporation)
- November 5, 2020 - Vulnerability disclosed publicly

Conclusion

Optiv disclosed a vulnerability in the “blackduck” package on PyPI that was remediated by Synopsys. This vulnerability is a reminder that every bit of code used by an organization is attack surface, even the tools that are meant to aid in securing the organization’s applications. When integrating these tools into your CI/CD pipelines, be sure to take the time to properly secure the integration.

References

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-27589
- https://github.com/blackducksoftware/hub-rest-api-python/commit/273b27d0de1004389dd8cf43c40b1197c787e7cd
- https://pypi.org/project/blackduck/
- https://github.com/blackducksoftware/hub-rest-api-python
- https://requests.readthedocs.io/en/latest/api/#requests.request
- https://github.com/blackducksoftware/bd-offline-scanning-solution/blob/9dd2e13d5ba987258aa5a3fbb13632ff862db56b/upload_scans.py#L102

By: James Otten

SENIOR SECURITY CONSULTANT | OPTIV

James is a senior security consultant with Optiv’s Threat Management team. In this role, he specializes in source code review, web application security and API security.

Share:

VULNERABILITIES THREAT RED TEAM SOURCE ZERO® CVE

Netwrix Account Lockout Examiner 4.1 Disclosure Vulnerability

August 13, 2020

Netwrix Account Lockout Examiner (versions prior to 5.1) allows an unauthenticated remote adversary to capture the NTLMv1/v2 challenge response....

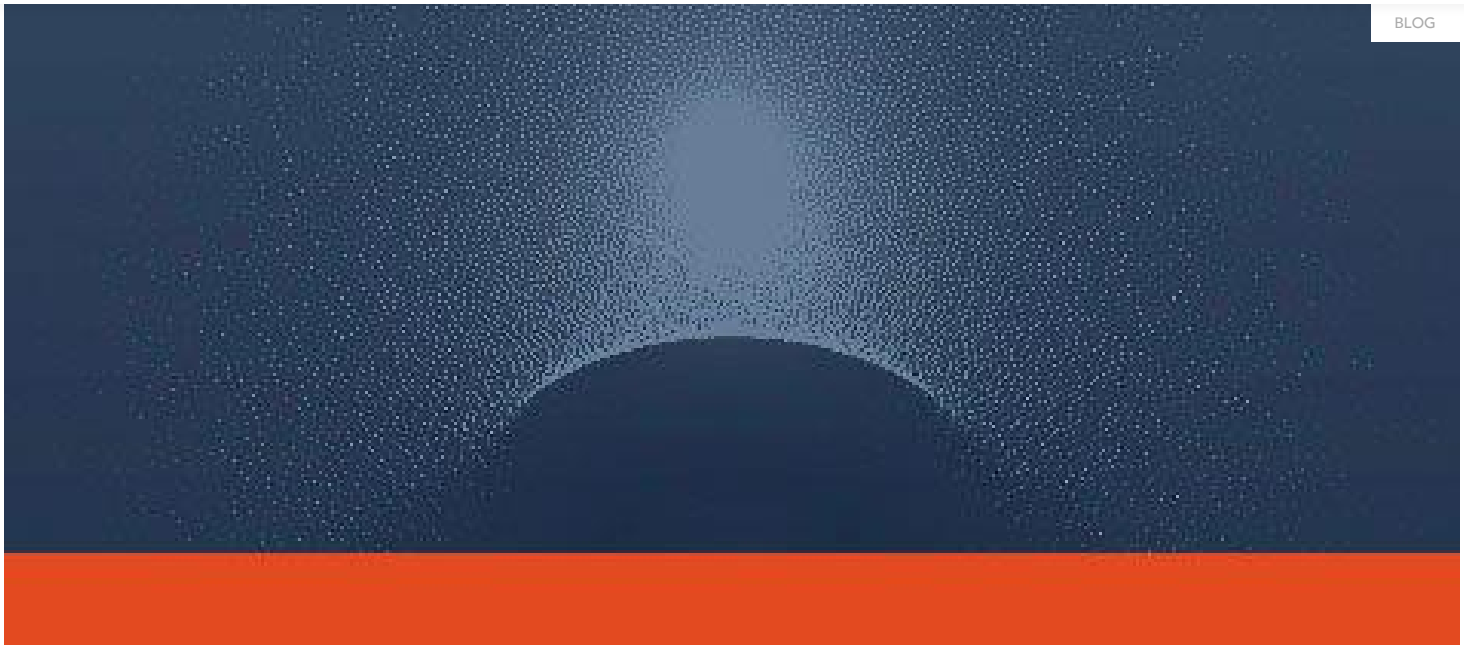
See Details

Optiv's REST API "Goat"

July 10, 2020

Optiv is releasing REST API Goat, a vulnerable API, to help boost AppSec skills.

See Details



BLOG

DEF CON 2020 Red Team Village Talk - Breaking The Attack Chain

September 02, 2020

Two experienced red teamers describe successful engagements and how to counter attack chains.

See Details

How Can We Help?

Let us know what you need, and we will have an Optiv professional contact you shortly.

First Name *

Last Name *

Company Email *

Company *

Job Title *

Phone *

Number of Employees *

Submit

SOLUTIONS

SERVICES

PARTNERS

INSIGHTS

ABOUT US

CAREERS

Secure greatness®