New issue

Jump to bottom

Python : Flask Path Traversal Vulnerability #669

Closed 1 of 2 tasks **porcupineyhairs** opened this issue on Apr 28 · 54 comments

Assignees

Labels invalid The Bug Slayer

porcupineyhairs commented on Apr 28

CVE(s) ID list

This is a placeholder issue. I plan on sending bulk PR's to approx 100 projects. I will add the CVE once the fixes are merged and identifiers are assigned.

All For One submission

#407

Details

TBA

Are you planning to discuss this vulnerability submission publicly? (Blog Post, social networks, etc).

- [x] Yes
- [ ] No

Blog post link

No response

🚀 2

This was referenced on Apr 28

Security Vulnerability Found noamezekiel/sphere#16

Closed

Security Vulnerability Found noamezekiel/sphere#17

Closed

Security Vulnerability Found pleomax00/flask-mongo-skel#1

Closed

jorgectf commented on Apr 28 • edited Contributor

👋 Just a detail. There's a typo in `This bug was found using [CodeQL by Github](here)` pointing to affected repo's `/issues/codeql.github.com` :)

👍 1

porcupineyhairs pushed a commit to porcupineyhairs/flask-mongo-skel that referenced this issue on Apr 28

`# Absolute Path Traversal due to incorrect use of send_file call ...` `0b1f1c2`

porcupineyhairs pushed a commit to porcupineyhairs/sphere that referenced this issue on Apr 28

`# Absolute Path Traversal due to incorrect use of send_file call ...` `184e0ee`

porcupineyhairs mentioned this issue on Apr 28

Security Vulnerability Found nlpweb/glance#2

Closed

porcupineyhairs pushed a commit to porcupineyhairs/glance that referenced this issue on Apr 28

`# Absolute Path Traversal due to incorrect use of send_file call ...` `25af4e7`

This was referenced on Apr 28

Security Vulnerability Found nlpweb/glance#4

Closed

Security Vulnerability Found olmax99/pyathenastack#7

Closed

porcupineyhairs pushed a commit to porcupineyhairs/pyathenastack that referenced this issue on Apr 28

`# Absolute Path Traversal due to incorrect use of send_file call ...` `45a0dbc`

This was referenced on Apr 28

Security Vulnerability Found nlpweb/glance#6

Closed

Security Vulnerability Found olmax99/pyathenastack#9

Closed

Security Vulnerability Found olmax99/helm-flask-celery#1

Closed

Security Vulnerability Found operatorequals/wormnest#7

Closed

Security Vulnerability Found orchest/orchest#906

Closed

Security Vulnerability Found onlaj/Piano-LED-Visualizer#350

Closed

Security Vulnerability Found nrlakin/homepage#1

Open

porcupineyhairs pushed a commit to porcupineyhairs/homepage that referenced this issue on Apr 28

`# Absolute Path Traversal due to incorrect use of send_file call ...` `c28c561`

This was referenced on Apr 28

Security Vulnerability Found piaoyunsoft/bt_lnmp#7

Open

Security Vulnerability Found pleomax00/flask-mongo-skel#3

Open

Security Vulnerability Found olmax99/pyathenastack#10

Open

Security Vulnerability Found nlpweb/glance#7

Open

Fix Path Traversal Vulnerability nlpweb/glance#8

Open

Fix Path Traversal Vulnerability olmax99/pyathenastack#11

Open

Security Vulnerability Found AFDudley/equanimity#2

Open

porcupineyhairs pushed a commit to porcupineyhairs/equanimity that referenced this issue on Apr 28

`# Absolute Path Traversal due to incorrect use of send_file call ...` `62ccc57`

127 hidden items

Load more...

porcupineyhairs commented on May 4 • edited ▾ Author

Hi,

So, all in all, I have been able to send in close to 92 reports. These are independent from the four CVE's referenced in #480.

Here are the details :

Sr. No.	Project	Current State	CVE ID	

1	ChaoticOnyx/OnyxForum	PR Merged.	GHSA-95jq-7f5x-v9x7	ChaoticOnyx/OnyxForu
2	clinical-genomics/scout	PR Merged.	CVE-2022-1554	https://www.huntr.dev/99e2-e4e17a81e600/
3	onlaj/Piano-LED-Visualizer	PR Merged.	GHSA-g78x-q3x8-r6m4	onlaj/Piano-LED-Visua
4	operatorequals/wormnest	PR Merged.	GHSA-v339-rw28-w5fr	operatorequals/wormn
5	orchest/orchest	PR Merged.	GHSA-j2rj-x939-977h	orchest/orchest#906
6	ChangeWeDer/BaiduWenkuSpider_flaskWeb	PR Merged.	CVE-2022-31504	
7	Microsoft/OLive	PR Merged.		https://www.huntr.dev/96cf-f7f81da10fff/
8	cheo0/MercadoEnLineaBack	PR Merged.	CVE-2022-31505	https://github.com/che
9	cmusatyalab/opendiamond	PR Merged.	CVE-2022-31506	cmusatyalab/opendian
10	ganga-devs/ganga	PR Merged.	CVE-2022-31507	ganga-devs/ganga#20
11	idayrus/evoting	PR Merged.	CVE-2022-31508	idayrus/evoting#1

12	iedadata/usap-dc-website	PR Merged.	CVE-2022-31509	https://www.huntr.dev/8883-d9ccf1f64024/
13	sergeKashkin/Simple-RAT	PR Merged.	CVE-2022-31510	sergeKashkin/Simple-R
14	AFDudley/equanimity	Fix Pending Merge.	CVE-2022-31511	AFDudley/equanimity#
15	Atom02/flask-mvc	Fix Pending Merge.	CVE-2022-31512	Atom02/flask-mvc#1
16	BolunHan/Krypton	Fix Pending Merge.	CVE-2022-31513	BolunHan/Krypton#1
17	Caoyongqi912/Fan_Platform	Fix Pending Merge.	CVE-2022-31514	https://github.com/Ca
18	Delor4/CarceresBE	Fix Pending Merge.	CVE-2022-31515	Delor4/CarceresBE#3
19	Harveyzyh/Python	Fix Pending Merge.	CVE-2022-31516	https://github.com/Ha
20	HolgerGraef/MSM	Fix Pending Merge.	CVE-2022-31517	HolgerGraef/MSM#12
21	JustAnotherSoftwareDeveloper/Python-Recipe-Database	Fix Pending Merge.	CVE-2022-31518	JustAnotherSoftwareDDatabase#9
22	Lukasavicus/WindMill	Fix Pending Merge.	CVE-2022-31519	Lukasavicus/WindMill#
23	Luxas98/logstash-management-api	Fix Pending Merge.	CVE-2022-31520	Luxas98/logstash-man

24	Niyaz-Mohamed/mosaic	Fix Pending Merge.	CVE-2022-31521	
25	NotVinay/karaokey	Fix Pending Merge.	CVE-2022-31522	NotVinay/karaokey#5
26	PaddlePaddle/Anakin	Fix Pending Merge.	CVE-2022-31523	PaddlePaddle/Anakin#
27	PureStorage-OpenConnect/swagger	Fix Pending Merge.	CVE-2022-31524	PureStorage-OpenCon
28	SummaLabs/DLS	Fix Pending Merge.	CVE-2022-31525	SummaLabs/DLS#4
29	ThundeRatz/ThunderDocs	Fix Pending Merge.	CVE-2022-31526	ThundeRatz/ThunderD
30	Wildog/flask-file-server	Fix Pending Merge.	CVE-2022-31527	Wildog/flask-file-serve
31	adobe/OSAS	Fix Pending Merge.		adobe/OSAS#16
32	bonn-activity-maps/bam_annotation_tool	Fix Pending Merge.	CVE-2022-31528	bonn-activity-maps/ba
33	cinemaproject/monorepo	Fix Pending Merge.	CVE-2022-31529	cinemaproject/monore
34	csm-aut/csm	Fix Pending Merge.	CVE-2022-31530	csm-aut/csm#104
35	dainst/cilantro	Fix Pending Merge.	CVE-2022-31531	dainst/cilantro#260

36	dankolbman/travel_blahg	Fix Pending Merge.	CVE-2022-31532	dankolbman/travel_bla
37	decentraminds/umbral	Fix Pending Merge.	CVE-2022-31533	decentraminds/umbra
38	echoleegroup/PythonWeb	Fix Pending Merge.	CVE-2022-31534	echoleegroup/PythonV
39	freefood89/Fishtank	Fix Pending Merge.	CVE-2022-31535	freefood89/Fishtank#4
40	jaygarza1982/ytdl-sync	Fix Pending Merge.	CVE-2022-31536	jaygarza1982/ytdl-sync
41	jmcginty15/Solar-system-simulator	Fix Pending Merge.	CVE-2022-31537	jmcginty15/Solar-syste
42	joaopedro-fg/mp-m08-interface	Fix Pending Merge.	CVE-2022-31538	joaopedro-fg/mp-m08
43	kotekan/kotekan	Fix Pending Merge.	CVE-2022-31539	kotekan/kotekan#1024
44	kumardeepak/hin-eng-preprocessing	Fix Pending Merge.	CVE-2022-31540	kumardeepak/hin-eng
45	lyubolp/Barry-Voice-Assistant	Fix Pending Merge.	CVE-2022-31541	lyubolp/Barry-Voice-As
46	mandoku/mdweb	Fix Pending Merge.	CVE-2022-31542	mandoku/mdweb#1
47	maxtortime/SetupBox	Fix Pending Merge.	CVE-2022-31543	maxtortime/SetupBox#

48	meerstein/rbtm	Fix Pending Merge.	CVE-2022-31544	meerstein/rbtm#47
49	ml-inory/ModelConverter	Fix Pending Merge.	CVE-2022-31545	ml-inory/ModelConver
50	nlpweb/glance	Fix Pending Merge.	CVE-2022-31546	nlpweb/glance#7
51	noamezekiel/sphere	Fix Pending Merge.	CVE-2022-31547	noamezekiel/sphere#1
52	nrlakin/homepage	Fix Pending Merge.	CVE-2022-31548	nrlakin/homepage#1
53	olmax99/helm-flask-celery	Fix Pending Merge.	CVE-2022-31549	olmax99/helm-flask-ce
54	olmax99/pyathenastack	Fix Pending Merge.	CVE-2022-31550	olmax99/pyathenastac
55	pleomax00/flask-mongo-skel	Fix Pending Merge.	CVE-2022-31551	pleomax00/flask-mong
56	project-anuvaad/anuvaad-corpus	Fix Pending Merge.	CVE-2022-31552	project-anuvaad/anuva
57	rainsoupah/sleep-learner	Fix Pending Merge.	CVE-2022-31553	rainsoupah/sleep-learn
58	rohitnayak/movie-review-sentiment-analysis	Fix Pending Merge.	CVE-2022-31554	rohitnayak/movie-revi
59	romain20100/nursequest	Fix Pending Merge.	CVE-2022-31555	sylann/nursequest#6

60	rusyasoft/TrainEnergyServer	Fix Pending Merge.	CVE-2022-31556	rusyasoft/TrainEnergyS
61	seveas/golem	Fix Pending Merge.	CVE-2022-31557	seveas/golem#11
62	tooxie/shiva-server	Fix Pending Merge.	CVE-2022-31558	tooxie/shiva-server#18
63	tsileo/flask-yeoman	Fix Pending Merge.	CVE-2022-31559	tsileo/flask-yeoman#2
64	uncleYiba/photo_tag	Fix Pending Merge.	CVE-2022-31560	uncleYiba/photo_tag#
65	varijkapil13/Sphere_ImageBackend	Fix Pending Merge.	CVE-2022-31561	varijkapil13/Sphere_Im
66	waveyan/internshipsystem	Fix Pending Merge.	CVE-2022-31562	waveyan/internshipsys
67	whmacmac/vprj	Fix Pending Merge.	CVE-2022-31563	whmacmac/vprj#2
68	woduq1414/munhak-moa	Fix Pending Merge.	CVE-2022-31564	woduq1414/munhak-r
69	yogson/syrabond	Fix Pending Merge.	CVE-2022-31565	https://github.com/yo
70	DSAB-local/DSAB	Vulnerability reported.	CVE-2022-31566	DSAB-local/DSAB#2
71	DSABenchmark/DSAB	Vulnerability reported.	CVE-2022-31567	DSABenchmark/DSAB#

72	Rexians/rex-web	Vulnerability reported.	CVE-2022-31568	Rexians/rex-web#5
73	RipudamanKaushikDal/projects	Vulnerability reported.	CVE-2022-31569	RipudamanKaushikDal
74	adriankoczuruek/ceneo-web-scrapper	Vulnerability reported.	CVE-2022-31570	https://www.huntr.dev/ab25-e02ec2b1a2ae/
75	akashtalole/python-flask-restful-api	Vulnerability reported.	CVE-2022-31571	akashtalole/python-fla
76	ceee-vip/cockybook	Vulnerability reported.	CVE-2022-31572	ceee-vip/cockybook#1
77	chainer/chainerrl-visualizer	Vulnerability reported.	CVE-2022-31573	chainer/chainerrl-visua
78	deepaliupadhyay/RealEstate	Vulnerability reported.	CVE-2022-31574	deepaliupadhyay/Reall
79	duducosmos/livro_python	Vulnerability reported.	CVE-2022-31575	duducosmos/livro_pytl
80	heidi-luong1109/shackerpanel	Vulnerability reported.	CVE-2022-31576	heidi-luong1109/shack
81	longmaoteamtf/audio_aligner_app	Vulnerability reported.	CVE-2022-31577	https://www.huntr.dev/85c7-f5f56f3676fc/
82	piaoyunsoft/bt_lnmp	Vulnerability reported.	CVE-2022-31578	piaoyunsoft/bt_lnmp#
83	ralphjzhang/iasset	Vulnerability reported.	CVE-2022-31579	https://github.com/ralp

84	sanojtharindu/caretakerr-api	Vulnerability reported.	CVE-2022-31580	sanojtharindu/caretake
85	scorelab/OpenMF	Vulnerability reported.	CVE-2022-31581	scorelab/OpenMF#263
86	shaolo1/VideoServer	Vulnerability reported.	CVE-2022-31582	shaolo1/VideoServer#
87	sravaniboinepelli/AutomatedQuizEval	Vulnerability reported.	CVE-2022-31583	sravaniboinepelli/Auto
88	stonethree/s3label	Vulnerability reported.	CVE-2022-31584	stonethree/s3label#10
89	umeshpatil-dev/Home__internet	Vulnerability reported.	CVE-2022-31585	umeshpatil-dev/Home
90	unizar-30226-2019-06/ChangePop-Back	Vulnerability reported.	CVE-2022-31586	unizar-30226-2019-06/
91	yuriyouzhou/KG-fashion-chatbot	Vulnerability reported.	CVE-2022-31587	yuriyouzhou/KG-fashic
92	zippies/testplatform	Vulnerability reported.	CVE-2022-31588	https://www.huntr.dev/ 8e13-6249be5ec005/

39 hidden items

Load more...

dependency checks going on, and dependency check failures require us to investigate and resolve such issues before we can merge code and produce releases. So we have multiple projects in need of suppressions now, which adds further maintenance burdens down the line.

I understand you may have little/no control over the CPE/mask that is assigned, and that there may be cases where tooling like DependencyCheck causes further issues, but I absolutely believe flagging a bunch of random people's side projects/test code with security vulnerabilities is at best an immense waste of time and resources of everyone involved, and at worst amounts to abuse of these reporting systems.

Where does it stop? Some random person's bash script which has a `rm -rf /` in it because he thought it would be fun to do some testing and happened to commit it to GitHub?

👍 8

ascopes commented on Jul 19

> Matching nothing is an improvement over matching everything (but still not much use). IMO - issuing CVEs for things that aren't consumable by anyone or anything else is not much use and just wasted noise. We will have to agree to disagree (or see what happens)

As much as I appreciate the fact that anyone can open a CVE, I totally agree with this. When creating a CVE, one of the things to consider is the impact of the vulnerability. If this is effectively a personal project that is not being used elsewhere, the impact is minimal. Letting the entire world know via a blanket CVE does not provide reasonable benefit to the development community.

> I understand you may have little/no control over the CPE/mask that is assigned, and that there may be cases where tooling like DependencyCheck causes further issues, but I absolutely believe flagging a bunch of random people's side projects/test code with security vulnerabilities is at best an immense waste of time and resources of everyone involved, and at worst amounts to abuse of these reporting systems.

It appears the CVE has been marked as rejected (https://nvd.nist.gov/vuln/detail/CVE-2022-31569) which is a flag that this CVE was potentially inappropriate. The given reason was:

```
** REJECT ** DO NOT USE THIS CANDIDATE NUMBER. ConsultIDs: none. Reason: This candidate was
withdrawn by its CNA. Further investigation showed that no specific affected product had been
identified. Notes: none.
```

👍 1 🎉 4

> which is a flag that this CVE was potentially inappropriate

Outside of the one that caused the issue, most all of the CVEs here on #669 (comment) are inappropate for the same reason. I'll be submitting those for re-evalutation as well. What a mess!

👍 2

marcelstoer commented on Jul 19 • edited

~~Instead of bulk-submitting CVEs IMO you should have bulk-submitted GitHub issue for the affected projects.~~
Ok, I see you did.

@skavanagh I also reached out to NIST asking to reject CVE-2022-31514.

👍 2

porcupineyhairs commented on Jul 19 Author

This thread has been blowing up quite a bit. This is most probably going to be my last and final comment for the issue at hand.

The GHSL team along with independent security researchers like me use tools like CodeQL to find security vulnerabilities in open source code. We try and recognise commonly occuring vulnerable patterns in codebases to prevent them from reoccuring. In this case, by perforing variant analysis, I found path traversal vulnerabilities in multiple projects. These were reported to their respective maintainers. Those who chose to fix them fixed them, those who didn't didn't. The entire reporting process was semi-automated and maintainers were given enough time before a disclosure was made. With each report, a CVE request was made so as to uniquely identify the issue. Now, CVE identifiers are by a variety of stakeholders for a variety of purposes; be it developers, their users, or security researchers. Sometimes, CVE records are used by devs to inform their users of a vulnerability, sometimes the records are used by authors of tools such as `dependency-check` to scan dependencies for known vulnerabilities and other times, they are used by folks like me to further our research. The CVE record list is not a text book you read page by page or record by record. The CVE project aims to catalog every cybersecurity vulnerability there is. They try to be as broad as possible when acccepting a new record. Given that the bug at hand, is a security vulnerability a CVE can be issued for it. There is no scarcity of record numbers and there's no real harm in registering one. In this case, I found a valid vulnerability and responsibly disclosed it. That's it. Fullstop. There's nothing more to it. All I did was share a link to this comment and requested a CVE. They did the rest. All those CPE identifiers, that not so helpful, probably script generated description for each individual CVE is all their doing. I don't have any role to play in it.

failing builds is a bug in the `OWASP dependency-check` project. Please file a bug report with that project. To repeat, The builds are not failing because I am requesting CVE ids, they fail because the tooling you use is buggy. If your tool can't figure out the difference between libs like `jakarta-annotation-api`, `JUnit 5` and `Fan_Platform`, I can't do much about it. Please check the issue tracker for that tool, there are multiple CVEs causing dependency checks to fail, not all of them are mine. Now coming to the second issue, as to whether MITRE and other CNA's should or shouldn't try and catalog every vulnerability there exists, all I can say is this issue is not the place for having philosophical discussions and hence, this conversation be taken somewhere else.

👎 8

marcelstoer commented on Jul 19

> The GHSL team along with independent security researchers like me use tools like CodeQL to find security vulnerabilities in open source code.

👍

> We try and recognise commonly occuring vulnerable patterns in codebases to prevent them from reoccuring.

👍

> In this case, by perforing variant analysis, I found path traversal vulnerabilities in multiple projects. These were reported to their respective maintainers. Those who chose to fix them fixed them, those who didn't didn't.

👍

> The entire reporting process was semi-automated and maintainers were given enough time before a disclosure was made.

👍

> With each report, a CVE request was made so as to uniquely identify the issue.

👎 Projects that have no license, no vendor, no version, no release, no package, or no build do not deserve to have CVEs IMO. I feel responsible security researchers should understand the - well, responsibility - they have.

> There is no scarcity of record numbers and there's no real harm in registering one.

👎 I disagree, it's just keeping systems and people busy needlessly. In some cases the amount of waste is significant.

Yes and no. By reporting CVEs that shouldn't exist in the first place you triggered it. Of course you are not responsible for potential bugs in the ODC project (jeremylong/DependencyCheck!4671). I feel that most people engaged in this discussion here do understand that perfectly well.

👍 7

skavanagh commented on Jul 19 • edited

> First, the issue of failing builds, folks who are here blaming me for their failing builds should understand that I am not the reason for their failed builds. The reason behind their failing builds is a bug in the OWASP dependency-check project. Please file a bug report with that project. To repeat, The builds are not failing because I am requesting CVE ids, they fail because the tooling you use is buggy.

OWASP dependency-check project is a best effort tool and only as good as the data it utilizes. It breaks up keywords from the vendor, project, and all the fields in the CPE to find matches. The more accurate the data is for the CVE, the more accurate the tool is (other tools too). We all deal with when these tools fail b/c it flags things it shouldn't, but usually the false-positives are against valid CVEs. I'm going to have NIST review all your CVEs. You filed junk data to the in NIST database - PEROID! It would be nice if you realized the mistake and help clean it up, but I guess not 🤷

👍 2

jorgectf commented on Jul 19 Contributor

Hi all,

Thank you all for bringing this information to our attention. It seems like there was an error assigning the CPEs for some of the CVEs associated with this Bug Slayer submission. We have already conveyed the issue to NIST.

As for the applicability of these CVEs, we do not encourage participants to ask maintainers to request CVEs for small, academic, or personal projects as stated in our bug bounty rules:

> Please note that projects which purposely include a vulnerability pattern for testing purposes, projects that are inactive in the last year, and student or academic projects, are considered out of scope.

And therefore, we will not take into account the CVEs assigned to projects matching those conditions.

We are and will be listening to your concerns and suggestions to improve the program and avoid future similar situations.

Thank you.

ascopes commented on Jul 19 • edited

Not wanting to argue with anyone or cause any problems, but I think it is worth adding a couple of points to what has already been said here, a little further up the thread. I appreciate both sides have valid arguments here, but there are a couple of things I disagree with.

> The CVE record list is not a text book you read page by page or record by record. The CVE project aims to catalog every cybersecurity vulnerability there is.

The main problem with this mindset is that if you want to document every single security vulnerability there is, you are going to end up with hundreds of thousands of vaguely labelled CVEs from projects made by students who have use-after-free vulnerabilities in their coursework for University. You are going to have vulnerabilities in random proofs of concept for tests. You are going to have CVEs for dummy hello world applications where people hardcode a fake password or dont sanitise a SQL input, or don't encode a URL path safely. All things that in a deployed system would be critical, but in a proof of concept or test, do not actually have any significance.

From this, a situation will arise where we are going to have so much noise and so many CVEs with vague CPEs that it will make the database completely unusable in enterprise and open source scenarios. This limits the use of any of these reports where this database is used for continuous integration and vulnerability detection to prevent human error.

This will just discourage the use of the NVE database entirely. The end result will be a far less secure cyberspace.

The purpose of a CVE is to evaluate the impact of a vulnerability (hence the 10 point rating system). A vulnerability that impacts, say, React Native, is going to naturally have a wider imapct than a hello world application that exists for purely symbolic purposes. With additional noise and misclassifications, it becomes increasingly difficult to accurately determine that impact. In the same way that it is far harder to hear someone talk if everyone in the room is shouting.

> If your tool can't figure out the difference between libs like jakarta-annotation-api, JUnit 5 and Fan_Platform

I think you have misunderstood the issue here a little. The issue is that it is somewhat difficult to determine what the library is when the CPE marks the name as `*`. Even for myself, it is not clear what that CVE is actually for. I still don't fully understand what library it is targeting. Even with the word `project` in it, it does not provide any meaningful information. Those CPEs are meant to be machine readable, which is why many sites print them as monospace.

I appreciate it is not your fault this was misconfigured, but it is worth taking into appreciation when considering how the thread has blown up.

Also worth noting some of the listed projects have had zero activity since 2015, so raising a CVE is going to have little effect on amending that code base. At least one of those I clicked on does not even exist.

skavanagh commented on Jul 19

I will be asking NIST to re-evaluate the following CVEs based on the justification below. There were some CVEs that were legit and should have been requested, but most weren't. These CVEs should absolutely not have been requested (or even approved) as there are no impacted products. We trip up on enough false-positives on valid CVEs. PRs are fine, but not CVEs.

CVE	JUSTIFICATION
CVE-2022-31504	https://github.com/ChangeWeDer/BaiduWenkuSpider_flaskWeb has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31505	https://github.com/cheo0/MercadoEnLineaBack This was a personal repository, but no longer exists. Regardless it has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31509	https://github.com/iedadata/usap-dc-website This is the website for USAP-DC and specific to that use case. It has no license or package that is consumable by another party.
CVE-2022-31510	https://github.com/sergeKashkin/Simple-RAT has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31511	https://github.com/AFDudley/equanimity has no usable license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31512	https://github.com/Atom02/flask-mvc has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31513	https://github.com/BolunHan/Krypton has a MIT license, but still no vendor, no version, no release, no package, no build, and is thus not consumable by another party (it's someone's personal repository).
CVE-2022-31514	https://github.com/Caoyongqi912/Fan_Platform has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31515	https://github.com/Delor4/CarceresBE has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.

2022-31516	Regardless it has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31518	https://github.com/JustAnotherSoftwareDeveloper/Python-Recipe-Database has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31520	https://github.com/Luxas98/logstash-management-api has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31521	https://github.com/Niyaz-Mohamed/Mosaic has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31526	https://github.com/ThundeRatz/ThunderDocs has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31527	https://github.com/Wildog/flask-file-server has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31528	https://github.com/bonn-activity-maps/bam_annotation_tool has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31532	https://github.com/dankolbman/travel_blahg has a MIT license, but still no vendor, no version, no release, no package, no build, and is thus not consumable by another party (it's someone's personal repository).
CVE-2022-31533	https://github.com/decentraminds/umbral has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31534	https://github.com/echoleegroup/PythonWeb has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31535	https://github.com/freefood89/Fishtank has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31536	https://github.com/jaygarza1982/ytdl-sync has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.

2022-31537	https://github.com/jmcginty15/Solar-system-simulator has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31538	https://github.com/joaopedro-fg/mp-m08-interface has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31540	https://github.com/kumardeepak/hin-eng-preprocessing has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31544	https://github.com/meerstein/rbtm has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31545	https://github.com/ml-inory/ModelConverter has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31546	https://github.com/nlpweb/glance has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31547	https://github.com/noamezekiel/sphere has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31548	https://github.com/nrlakin/homepage has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31551	https://github.com/pleomax00/flask-mongo-skel has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31552	https://github.com/project-anuvaad/anuvaad-corpus has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31553	https://github.com/rainsoupah/sleep-learner has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31554	https://github.com/rohitnayak/movie-review-sentiment-analysis has a MIT license, but still no vendor, no version, no release, no package, no build, and is thus not consumable by another party (it's someone's personal repository).

2022-31555	no release, no package, no build, and is thus not consumable by another party (it's someone's personal repository).
CVE-2022-31556	https://github.com/rusyasoft/TrainEnergyServer has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31557	https://github.com/seveas/golem has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31559	https://github.com/tsileo/flask-yeoman has a MIT license, but still no vendor, no version, no release, no package, no build, and is thus not consumable by another party (it's someone's personal repository).
CVE-2022-31560	https://github.com/uncleYiba/photo_tag has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31561	https://github.com/varijkapil13/Sphere_ImageBackend has a BSD license, but still no vendor, no version, no release, no package, no build, and is thus not consumable by another party (it's someone's personal repository).
CVE-2022-31562	https://github.com/waveyan/internshipsystem has a MIT license, but still no vendor, no version, no release, no package, no build, and is thus not consumable by another party (it's someone's personal repository for a student internship).
CVE-2022-31563	https://github.com/whmacmac/vprj has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31564	https://github.com/woduq1414/munhak-moa has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31565	https://github.com/yogson/syrabond has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31566	https://github.com/DSAB-local/DSAB has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31567	https://github.com/DSABenchmark/DSAB has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.

2022-31568	https://github.com/Rexians/rex-web This is the website for Rexians Community and specific to that usecase. It has no license or package that is consumable by another party.
CVE-2022-31570	https://github.com/adriankoczuruek/ceneo-web-scrapper has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31571	https://github.com/akashtalole/python-flask-restful-api has a MIT license, but still no vendor, no version, no release, no package, no build, and is thus not consumable by another party (it's someone's personal repository).
CVE-2022-31572	https://github.com/ceee-vip/cockybook has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31574	https://github.com/deepaliupadhyay/RealEstate has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31575	https://github.com/duducosmos/livro_python has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31576	https://github.com/heidi-luong1109/shackerpanel has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31577	https://github.com/longmaoteamtf/audio_aligner_app has a MIT license, but still no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31578	https://github.com/piaoyunsoft/bt_lnmp has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31579	https://github.com/ralphjzhang/iasset This was a personal repository, but no longer exists. Regardless it has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31582	https://github.com/shaolo1/VideoServer, has a GPL license, but still no vendor, no version, no release, no package, no build, and is thus not consumable by another party (it's someone's personal repository).
CVE-2022-31583	https://github.com/sravaniboinepelli/AutomatedQuizEval has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.

2022-31585	https://github.com/umeshpatil-dev/Home_Internet has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31587	https://github.com/yuriyouzhou/KG-fashion-chatbot has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.
CVE-2022-31588	https://github.com/zippies/testplatform has no license, no vendor, no version, no release, no package, no build, and is thus not consumable by another party.

porcupineyhairs commented on Jul 19 — Author

Here's what MITRE has to say about all of this

```
Thank you for contacting us. We are primarily concerned with the
amount of time required to debate CVE Records for products that are
not widely used. Regardless of whether something is a "valid
vulnerability in a valid Github repo" it is possible that nobody will
ever use the affected code. We cannot devote a lot of time to the
optimal resolution of debates about software that has very few users
and/or would be difficult to incorporate into other projects.

CVE Records need to be about software products, not solely about
Github repos. By far, the most common case is that one repo is used
for one software product. However, as you can see,

   https://github.com/RipudamanKaushikDal/projects

is about many unrelated software-development efforts by the same
person. The information in your original CVE ID request pointed only
to https://github.com/github/securitylab/issues/669 and did not point
to the https://github.com/RipudamanKaushikDal/projects/issues/21 issue
that said the vulnerability was in something called
image_search_gallery. This makes the CVE ID request and the CVE Record
completely invalid, and it is removed from the main CVE List. We do
keep it in our GitHub repo forever:


https://github.com/CVEProject/cvelist/blob/c791c48acd4ec8c16b331cc745e98f3118d21568/2022/31xxx/CVE-
2022-31569.json

in case someone needs it. However,
https://github.com/RipudamanKaushikDal/projects seems to be about
copying code to GitHub for personal learning without any intention to
make the code usable by other people.

If you need a CVE ID for image_search_gallery, we can provide one, but
```

```
Fan_Platform is a completely different case. Indeed, earlier today
someone wanted it removed, but we did not remove it, Instead we told
them that their objections were not sufficient. We looked at it and
decided:

  1. There might be a bug in
  https://github.com/jeremylong/DependencyCheck that causes Fan_platform
  to match ^pkg:maven/org\.junit\.platform/junit\-platform - in other
  words, the presence of the "platform" substring might be enough for
  DependencyCheck to guess that there is a dependency

  2. There has been an assertion of "no license" in a situation where
  multiple GitHub users have been contributing to the codebase. Even if
  you do not know what the license is, the continuing activity by
  multiple persons makes it reasonable to expect that they have arranged
  among themselves to allow use of the software.

Again, because of time constraints, we will not necessarily send any
responses or notifications about additional CVE Records that people
want rejected. Your work is still a valid application of the CodeQL
tool even if the GitHub repos aren't for widely used software, and your
work can potentially improve the process by which other persons use
CodeQL in the future. Thank you for your contribution.
```

porcupineyhairs commented on Jul 19 — Author

I can see CVE-2022-31569 has been removed. MITRE says I can seek a new CVE with the correct sub-project name but they don't encourage this. So I am not going to apply for a CVE that particular project again. As for the disputes to all other CVE's are considered. I think they are effectively addressed by the message above. Everyone visiting this thread for addressing their failed builds can take up the issue of the false positive with their respective tool maintainers.

With that said, **@jorgectf**, the bounty application still remains in place. So far I have 13 projects and 2 independent forks which have addressed the vulnerability. Of these, `Microsoft/Olive` has merged the fix but is yet to issue a CVE. I received a message from them late last week that they are still working on this and would get back soon. `cheo0/MercadoEnLineaBack` had merged the fix but now the repo has been deleted. I do have a clone `porcupineyhairs/MercadoEnLineaBack` which has been included in the LGTM run I shared earlier. As for the forks, `scaraude/Piano-LED-Visualizer` and `Nbobito/Piano-LED-Visualizer`, these are forks of `onlaj/Piano-LED-Visualizer` but have divereged from the original repo. They should be treated as separate projects but I haven't sought a CVE for them and I haven't included them in the LGTM run.

The impact metrics for of these vulnerabilities remains the exactly as we discussed here

👍 1 ❤️ 3

I'll appeal to NIST directly on the CVEs in question, but to address the points made

> There has been an assertion of "no license" in a situation where multiple GitHub users have been contributing to the codebase. Even if you do not know what the license is, the continuing activity by multiple persons makes it reasonable to expect that they have arranged among themselves to allow use of the software.

A lack of licensing would mean a library or project is less likely to be used elsewhere. A lack of a release is even less likely, A build even more. And on top of all, if it hasn't even been published or versioned - I think it's safe to question if anything is really impacted. Do any of these CVEs have any sort of product or service behind them that is vulnerable?

> There might be a bug in https://github.com/jeremylong/DependencyCheck that causes Fan_platform to match ^pkg:maven/org.junit.platform/junit-platform - in other words, the presence of the "platform" substring might be enough for DependencyCheck to guess that there is a dependency'

Dependency check (and other tools) have to do this on purpose. They break up the keywords in the CPE to identify possible vulnerable dependencies. If the NIST data could be used to perfectly identify a vulnerable dependency then there wouldn't be the need to do this and all would be well. This "bug" sometimes accounts for the very problem caused by the negligent submission CVEs. Bad data in = Bad data out. You can't build good tools around bad data. Not saying the NIST data is bad, it's very good! We find legit vulnerabilities all the time, but we need to keep it that way. Submitting CVEs for random github projects does not serve that purpose. The world does not need to know about a vulnerability in Uncle Yiba's photo_tag project - CVE-2022-31560. It takes manpower and computing power to ensure things are identified properly. It's not cheap!

👍 5

 **marcelstoer** mentioned this issue on Jul 21

Fix #4671 jeremylong/DependencyCheck#4688

Merged

 **skavanagh** mentioned this issue on Jul 23

Reject Python pet projects CVEs CVEProject/cvelist#6623

Closed

xcorail commented on Jul 28 Contributor

assessing the findings we have determined this submission is not eligible for a reward under the Bug Bounty program, because the security vulnerabilities have been disclosed publicly, in public Issues and Pull Requests.

The Bug Slayer program requires that you disclose the vulnerabilities following a **coordinated disclosure** process. Here are the extracts of our rules that mention this:

> To be eligible for a bounty, you must first coordinate disclosure and fix of the vulnerabilities with the maintainers of the projects

> The CVEs must correspond to vulnerabilities that have been disclosed and fixed via coordinated disclosure with the maintainers.

We know that private collaboration between security researchers and maintainers is sometimes difficult, but that is precisely the point of this program. As stated in our rules:

> The goal of this program is to incentivize the collaboration with maintainers to fix vulnerabilities.

Even though this submission was assessed as being ineligible for the bounty program, we appreciate your efforts finding and fixing security vulnerabilities at scale in open source projects, and would like to offer you a thank you reward.

Best regards and happy hacking!

 **xcorail** closed this as completed on Jul 28

---

ghsecuritylab commented on Jul 28 — Collaborator

Your submission is now in status **Closed**.

For information, the evaluation workflow is the following:
Initial triage > Test run > Results analysis > Query review > Final decision > Pay > Closed

porcupineyhairs commented on Aug 4 — Author

Okay, so `@xcorail` and I discussed this on Slack the other day. I am posting a brief below to keep track of the conversation. Here's what I understand so far.

GHSL wants to promote responsible behaviour and hence requests that a researcher only make a Coordinated Vulnerability Disclosure(CVD). In this case, GHSL deems that by making a public issue and a public PR with a fix, I have violated CVD norms. Instead of filing public issues and public PR's, I should have contacted each maintainer individually and requested them to either fix the vulnerability, provide them a private patch or at least seek prior permission before submitting the PR.

Reporting, Validation and Triage, Remediation, Public Awareness and Deployment. It explicitly says that a stakeholder may be responsible for more than one phase. In this case, since everthing till the Remediation phase was handled by me. I discovered the underlying issue, validated the vulnerability, drafted a report and even created a patch; I claim that there is no real adversary advantage resulting from my acts. None of the projects were, even for a few minutes, left without a patch.

I claim, since the vendor for open source projects is typically the owner of the repo and the repository's contributors, by the virtue of me submitting a tested patch, I, for the purposes of CVD, can act as a vendor. `@xcorail` does not agree with this. They say, the vendor should only be the owner or a previous contributor. I can't deem myself as a vendor and publish a patch.

I would also like to point out that my CodeQL patch was merged directly into an already existing stable query. So the vulnerabilities were already publicly visible on LGTM. In fact, walking through the LGTM alerts for the particular rule is what led me to those projects in the first place. While I claim above that I carried out the reporting phase, it was infact LGTM which did it. The vulnerability was publicly shown on LGTM well before I attempted to patch them. My public Github issues/PRs don't make any new additions on that front. My PRs on the contrary aid in fixing them. I might have made the public disclosure which prompted the maintainers to act but I am not the first one to discuss the vulnerability publicly. LGTM did that before me by making alerts publicly visible.

By this decision, I believe GHSL is implying that no new alert on LGTM, resulting from a CodeQL stable query, or any additional patch on it, would ever be eligible for Bug Slayer. Is this intended?

😟 1

xcorail commented on Aug 4 Contributor

> I claim, since the vendor for open source projects is typically the owner of the repo and the repository's contributors, by the virtue of me submitting a tested patch, I, for the purposes of CVD, can act as a vendor. **@xcorail** does not agree with this. They say, the vendor should only be the owner or a previous contributor. I can't deem myself as a vendor and publish a patch.

Indeed I do not agree with you. The reporter cannot be a vendor. If you look at page 17 of the CERT guide that you mention, the vendor is clearly defined, and when they talk about open source they say `Many open source libraries are maintained by a single person or a small independent team; we still refer to these individuals and groups as vendors` which implies that in the open source case, the vendor is a maintainer or a group of maintainers of the projects (and not all contributors). You can totally publish a patch if you want, but you cannot consider yourself as a `vendor` in the documented CVD. The CVD is a **collaboration** between the reporter and the vendor, so the reporter cannot be the vendor.

> By this decision, I believe GHSL is implying that no new alert on LGTM, resulting from a CodeQL stable query, or any additional patch on it, would ever be eligible for Bug Slayer.

skavanagh commented on Aug 4

I'll just chime in here too.. b/c you are doing things so wrong.

> I discovered the underlying issue, validated the vulnerability, drafted a report and even created a patch

I don't see how you could have done this. Most of those "projects" don't have a viable application that you could have gotten up and running. Have you done a reproduction or exploited any of these "applications" that you have alerted the public about? Just b/c something shows in a SAST scan doesn't mean it's vulnerable. It's got to be explotible in the context of a running application.

> I claim, since the vendor for open source projects is typically the owner of the repo and the repository's contributors, by the virtue of me submitting a tested patch.

Yeah.. that's BS! You couldn't have "tested" the patch on a running "application" there are no applications to run on those CVEs you've opened. Maybe you've tested the remediation by itself, but you have not in the context of those projects. It's just scraps of code in various repositories.

IMO - Mitre has handled this wrong too. Lots of wrong everywhere.

👍 1

porcupineyhairs commented on Aug 4 Author

@xcorail

> The LGTM alerts are indeed visible to the public, but they are not yet verified and triaged. If you use one of these alerts, verify it, create a PoC for it, propose a patch or not,

CVD says verification and triage is done by the vendor. Provided that the vendor can reproduce the issue by reading the report, PoC, patch etc are not required by CVD. In this case, qhelp would typically suffice.

> disclose it privately following the CVD, and get the consent of the maintainer to create a public PR, then that would be eligible to our program.

The disclosure is already done. A report's a report regardless of who does it. A report even from a less trusted source is still processed the same way. Yes, during verification/triage the vendor may still decline it but I don't see any special exclusions for anyone there. I am trying to understand how a report from LGTM differs from a report by me on this front.

> In this case, GHSL deems that by making a public issue and a public PR with a fix, I have violated CVD norms. Instead of filing public issues and public PR's, I should have contacted each maintainer individually and requested them to either fix the vulnerability, provide them a private patch or at least seek prior permission before submitting the PR.

Isn't this the whole definition of responsible disclosure? You don't make dangerous information known to the world until actions have been taken to prevent it being exploited by falling into the wrong hands. By making an issue the center of attention by raising public paperwork for it, you are just holding up a big flashing sign saying "hey, I found an exploit for your application that is dangerous, come see".

If the exploit isn't dangerous or has very limited or no scope, it then questions the appropriateness of a CVE.

> The vulnerability was publicly shown on LGTM well before I attempted to patch them.

LGTM just shows the potential for a vulnerability to be present. It does not understand every edge case that exists to prove something is exploitable. LGTM is an advisory assistive tool, not a single source of truth.

LGTM is also an automated tool, it lacks the ability to exercise situational discretion in the same way you do as a human. This is important to distinguish.

I have seen applications that LGTM would raise as being vulnerable to CVEs that are just totally incorrect (one being Log4Shell), purely because a dependency has an optional binding to that library. The noise by false positives is always a deterrent for people looking for quick ways to exploit deployed software. LGTM can also be left misconfigured for some projects, which can add additional noise that an attacker would have to spend time and resources working around.

By raising issues and PRs before a fix has been evaluated and ideally implemented, you are effectively reducing the legwork needed by people scraping for exploits. Unlike LGTM, you are saying "I have proven that this is vulnerable", not just "this has the potential to be vulnerable based on other cases I know of".

For most Flask apps that are just hobby/pet projects, the impact is somewhat limited, but imagine if you did this for heartbleed, or spectre-like attacks. It would throw everyone into an immediate panic.

👍 1

porcupineyhairs commented on Aug 4 • edited ▾ Author

@ascopes

> By making an issue the center of attention by raising public paperwork for it, you are just holding up a big flashing sign saying "hey, I found an exploit for your application that is dangerous, come see".

> LGTM just shows the potential for a vulnerability to be present. It does not understand every edge case that exists to prove something is exploitable. LGTM is an advisory assistive tool, not a single source of truth.

In the coordinated vulnerability disclosure process, each report is to be assesed independently. The sources may be less trusted but that does not mean you disregard the report.

> I have seen applications that LGTM would raise as being vulnerable to CVEs that are just totally incorrect (one being Log4Shell), purely because a dependency has an optional binding to that library. The noise by false positives is always a deterrent for people looking for quick ways to exploit deployed software.

Agreed.

> By raising issues and PRs before a fix has been evaluated and ideally implemented, you are effectively reducing the legwork needed by people scraping for exploits. Unlike LGTM, you are saying "I have proven that this is vulnerable", not just "this has the potential to be vulnerable based on other cases I know of".

The fix are comparitively minor and all assessed manually by me. A report for "has the potential to be vulnerable" is still a report worth taking a look at according to CERT's CVD guidelines. Yes, one could have looked at my profile and found a very useful set of results but I don't know if there would be a meaningful impact in this case.

Basically, I am seeking a few clarifications on GHSL's bounty program. GHSL is rejecting my report saying I am making a public disclosure instead of a CVD. But the thing is if you strictly go by the CVD guidelines, then even LGTM alerts are treated as valid reports. If GHSL is disqualifying my report due to non-conformance with CVD guidelines, I would like to know if GHSL plans on disqualifing all cases where a public LGTM alert was shown as ineligible on similar grounds.

👎 1

ascopes commented on Aug 4 • edited

> In the coordinated vulnerability disclosure process

This is kind of missing my point. My point is not that it follows some defined process. My point is that it is a difference between speculation and confirmation.

> I sent in a public patch to fix bugs which were valid after manual testing.

That's the point, you explicitly declared and confirmed the vulnerability. Without doing that, it is not clear that the issue was actually a problem in the first place.

seeing the dependency list most of the time. LGTM is just linking that and the patterns for it. Neither are disclosing a confirmed issue.

It is a case of providing additional information that was not there before, either via clarity, reproductions, confirmation, or additional attack vectors.

Not speaking on behalf of the project, but that is my assumption as to why this is the case.

👍 2

xcorail commented on Aug 4 Contributor

> Basically, I am seeking a few clarifications on GHSL's bounty program. GHSL is rejecting my report saying I am making a public disclosure instead of a CVD. But the thing is if you strictly go by the CVD guidelines, then even LGTM alerts are treated as valid reports.

I don't agree. LGTM alerts are not `valid reports` . They are just results of a SAST tool, without any human additional analysis of whether the alert is exploitable or not.

> If GHSL is disqualifying my report due to non-conformance with CVD guidelines, I would like to know if GHSL plans on disqualifing all cases where a public LGTM alert was shown as ineligible on similar grounds.

I already answered this question above. If you transform one of these alerts into an actionable report, by adding your added value as a security researcher (examples from what **@ascopes** said: `either via clarity, reproductions, confirmation`), and disclose it privately to the maintainer, then it would be eligible.

👍 3

skavanagh commented on Aug 4

I use CodeQL on my projects, BTW! Love it!

❤️ 2

porcupineyhairs mentioned this issue on Aug 7

Go : PAM Authorization Bypass #686

Closed

2 tasks

NickHastings pushed a commit to NickHastings/ganga that referenced this issue on Sep 12

 `# Absolute Path Traversal due to incorrect use of send_file call (g…` ··· `8157aae`

Assignees

 **jorgectf**

Labels

invalid The Bug Slayer

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

15 participants