Lab: GitHub

Background

ACME Corp. is a technology-forward organization where GitHub is an approved and heavily used resource. Engineering, IT, and security teams regularly access public repositories from vendors and open-source communities such as cloud providers, operating system maintainers, and AI research organizations.

Recent analysis of the SCATTERED SPIDER campaign C0027 highlighted the threat actor's use of platforms like GitHub, file[.]io, and pastee[.]ee for tool downloads. This information, coupled with a webinar indicating that DPRK IT Workers frequently utilize GitHub, has prompted the security team to focus on *how* GitHub is being used, rather than whether it should be used at all. The team plans to initiate a targeted hunt to gather evidence and build a compelling business case for the CIO regarding necessary policy changes related to the platform.

Concern
During a recent internal review, the security team learned that threat actors increasingly abuse public GitHub repositories to host offensive tooling, malware source code, proof-of-concept exploits, and post-exploitation frameworks. These repositories often look indistinguishable from legitimate development activity when viewed at scale.

Because GitHub traffic is high-volume, business-critical, and broadly authorized, the IT and security teams are struggling to answer a simple but important question:

> *How do we identify risky or suspicious GitHub activity hidden inside a large amount of legitimate use (estimated 5,000+ daily GitHub transactions)?*

Blocking GitHub outright is not an option, and signature-based detections are insufficient since new repos are created daily. The team needs a threat-hunting approach that can surface anomalous or high-risk GitHub activity without disrupting the business.

Lab Objective

In this lab, you will analyze Zscaler Internet Access (ZIA) web proxy telemetry to:

- Differentiate benign repository access from suspicious activity
- Identify GitHub access patterns that may indicate:
    - Malware research
    - Unauthorized offensive tooling
    - Early-stage attacker tradecraft research

How the Lab Data Is Structured

Within the provided CSV lab files (lab_github.csv):

- The user accesses numerous legitimate GitHub repositories, including well-known organizations and projects (e.g., cloud providers, operating system vendors, AI tooling).
- Mixed into that traffic is one or more visits to a suspicious repository that does not immediately stand out unless you:
    - Pivot on repository name, path, or topic
    - Use least frequency of occurrence (LFO) or anomaly identification techniques
    - Apply threat-hunting intuition instead of binary allow/deny logic

Your task is to find the suspicious GitHub activity and explain why it matters. Use the table below to help you track observations and findings.

Date/Time	Username	URL	Analyst Notes

Most GitHub content follows a consistent URL structure:

`github.com/<owner>/<repository>/...`

`raw.githubusercontent.com/<owner>/<repository>/...`

The `owner/repository` portion uniquely identifies a project and stays the same regardless of whether a user is browsing code, viewing files, or downloading raw content. For example, the repository for this lab is:
`https://github.com/themikewylie/threat-hunting-lab-zia`

When threat hunting, grouping GitHub activity by `owner/repository` helps reduce noise and makes it easier to spot unusual or high-risk repositories hidden among large volumes of legitimate GitHub use. To illustrate this, run the following commands.

In Bash:

```
awk -F',' '$8 ~ /github/ { print $8 }' ./data/lab_github.csv | sort
| uniq -c | sort -nr
```

You'll notice that we successfully honed in on GitHub traffic, reducing by a factor of eight. However there are still 1,000+ transactions (Pro Tip: append `| wc -l` to count the rows) in our subset of data. In this example it was only one user's activity over a 30 minute period. In a large enterprise the subset of data would still be unmanageable.

To make the data more manageable, we can group and count by `owner/repository`.

```
awk -F',' 'NR>1 &&
match($8,/(github\.com|raw\.githubusercontent\.com)\/[^\/]+\/[^\/]+
/) {
  r=substr($8,RSTART,RLENGTH)
  sub(/^(github\.com|raw\.githubusercontent\.com)\//,"",r)
  print r
}' ./data/lab_github.csv | sort | uniq -c | sort -nr
```

This approach eliminates the noise of users browsing and accessing files/folders within a repository and groups them as a single row for our long-tail analysis. The results provide a handful of repositories for us to review. Do any of the results raise a red flag?

BONUS POINTS: Write a python script (or vibe code a tool) to take the unique GitHub URLs and gather context (e.g. is there a README page? How popular is the repo? How many stars and forks? What metadata can you gather to help provide context about the use of each distinct GitHub repo?).

Answering these questions will help you narrow down the outliers in the dataset.

Questions	**Answers**
How many GitHub transactions are in the dataset?	
What artifacts make a benign GitHub repo less risky?	
Which repos are most frequently interacted with?	
Which repos are least frequently interacted with?	

In threat hunting, we are constantly trying to make good huts automated and minimize the manual sifting of data so we can research and create new hunts. Checkout the `github_enrichment.ipynb` Jupyter Notebook in the `/notebooks/` directory.