AppSec: Origins to Innovations

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

THANK YOU!

- Mike McCabe
- Chrystina Nguyen
- ALL OF YOU! 👏👏👏

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Ken Johnson
@cktricky

Co-Host of Absolute AppSec
CTO of DryRun Security
BJJ Nerd

DRYRUN .SECURITY

Outline

- Origins
    - Timeline
    - Takeaways
- Innovations
    - AI
    - Potential Use Cases

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Origins

DRYRUN .SECURITY

Origins: Notable markers

1995 - 2001

***1970 Waterfall was released**

95 JavaScript released
98 SQL Injection
98 AppScan by Sanctum
01 Web Inspect by Spi Dynamics
01 Agile Manifesto
01 "Shift-Left-Testing" by Larry Smith
01 OWASP! Mark Curphey

2002-2011

02 Ounce labs
03 Fortify
06 Veracode
06 CheckMarx
08 Agile gains traction
09 DevOps (Patrick Debois & Andrew Shafer)
09 OPENSAMM (*2016 gained traction under OWASP)
10-ish DevSecOps infancy (won't really evolve until around 2014)

***Around this era, threat modeling starts to take shape**

2012-2016

12 Dependency Check
12 A9 OWASP Category introduces
13 RASP becomes a thing
13+ Josh Corman / Sonatype begin to focus on SCA in their product line
15 Synk launches (SCA)
15 Dev Training Gamified / Secure Code Warrior
16 - Jason Chan / Netflix discusses Paved Roads(paths)

2017-2024

19 Semgrep
19 CodeQL
20 ASPMs (Enso?)
21 Executive Order (SBOMs)
22 Co-Pilot
22-23 LLMs go mainstream
23 "Shift-Left" falling out of favor, "Shift Smart"
23 SCA Reachability Analysis
23 StrideGPT
24 Auto-remediation for Devs

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Takeaways - Tools

🛠️ Tools evolution

- DAST
- DAST + SAST
- SAST + SCA + RASP
- SAST + SCA + SBOM + ASPM
- SAST, SCA, SBOM, ASPM
- **EMERGING:** Auto-Threat Modeling, Auto-Remediation, Auto-Assistant (Co-Pilot)

🖥️ SAST overtook DAST, RASP/IAST still in play

🌶️Tools marketed at devs were really just the same security-expert tools re-packaged for CI/CD

⚙️We've been making the same types of tools… just slightly smarter and more of them

🚀**EMERGING:** More focus on customization and automating developers assistance

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Takeaways - Process

👷Prevention Evolution

- Originally… Testing
- Testing, Training
- Testing, Training, Threat modeling
- Testing, Training, Threat Modeling, Guard Rails / Paved Paths

🔍Testing Evolution

- Test at last stages of SDLC
- Test earlier in SLDC
- Test as software is being developed? (*I have thoughts here* 💅)

⚠️Threat Modeling Evolution

- Security did it
- Devs started doing it
- *…AI has entered the chat*

🧑‍🏫Training

- SCORM Compliant CBTs
- Gamified Security Training
- Gamified ++

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Takeaways - Strategies

👷Strategy Evolution

- Find Bugs
- Find Bugs + Manage Bug Tickets
- Focus on Prevention
- Mature and measure
- Find and Manage Bug Tickets…. **BUT NOW AT SCALE** BRAWNDO
- Educate, prevent, find/fix, manage risk at scale

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Innovations

DRYRUN
.SECURITY

Innovations - AI

- Progress goes by many names: AI, LLMs, NLP, ML, etc.
- What is it fundamentally good at?
    - Pattern Analysis
    - Text Summarization
    - Similarity Searches
- What is it NOT good at
    - Outputting the exact same word structure
    - Handling large amounts of context (so far)
    - Free-form analysis without heavy work/guidance (at scale, production level)

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Innovations

False claims:

- AI learn from your interactions with it
- AI is too unpredictable and inaccurate for any “serious” work
- AI can do it all and will replace us in the next few years
- AI can’t be used to identify vulnerabilities in software

True claims:

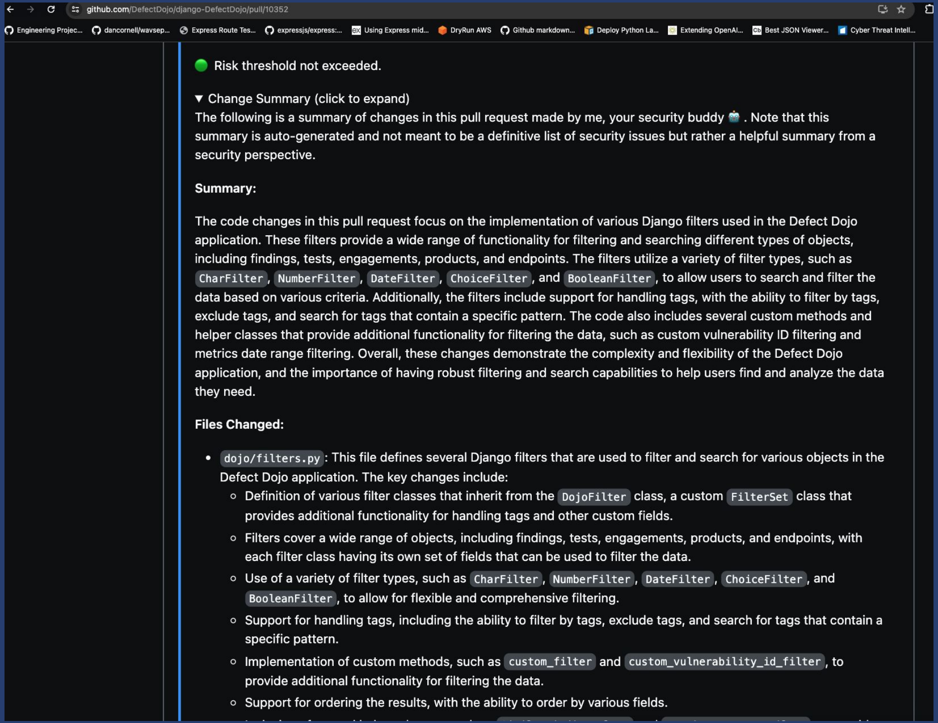
- OpenAI stores personal account interactions to train their model on later
- AI models are decreasing in size and increasing in performance at a rapid pace

Innovations

Enhancements to existing approaches

- Automated Design Reviews & Suggestions
- Automated Threat Modeling on individual changes / PRs
- Alerts on riskiest changes/PRs
- Alerts to the riskiest services in your organization
- Developer secure coding assistants
- Chat for first level triage of developer questions
- Automate training - new significant vulnerabilities placed into training content instantly
- Automated dependency updates that don’t suck
- Understand what is happening at a macro level in your organization

Innovations: Examples of Summarizing PR changes



🟢 Risk threshold not exceeded.

▼ Change Summary (click to expand)

The following is a summary of changes in this pull request made by me, your security buddy 🤖 . Note that this summary is auto-generated and not meant to be a definitive list of security issues but rather a helpful summary from a security perspective.

Summary:

The code changes in this pull request focus on the implementation of various Django filters used in the Defect Dojo application. These filters provide a wide range of functionality for filtering and searching different types of objects, including findings, tests, engagements, products, and endpoints. The filters utilize a variety of filter types, such as `CharFilter`, `NumberFilter`, `DateFilter`, `ChoiceFilter`, and `BooleanFilter`, to allow users to search and filter the data based on various criteria. Additionally, the filters include support for handling tags, with the ability to filter by tags, exclude tags, and search for tags that contain a specific pattern. The code also includes several custom methods and helper classes that provide additional functionality for filtering the data, such as custom vulnerability ID filtering and metrics date range filtering. Overall, these changes demonstrate the complexity and flexibility of the Defect Dojo application, and the importance of having robust filtering and search capabilities to help users find and analyze the data they need.

Files Changed:

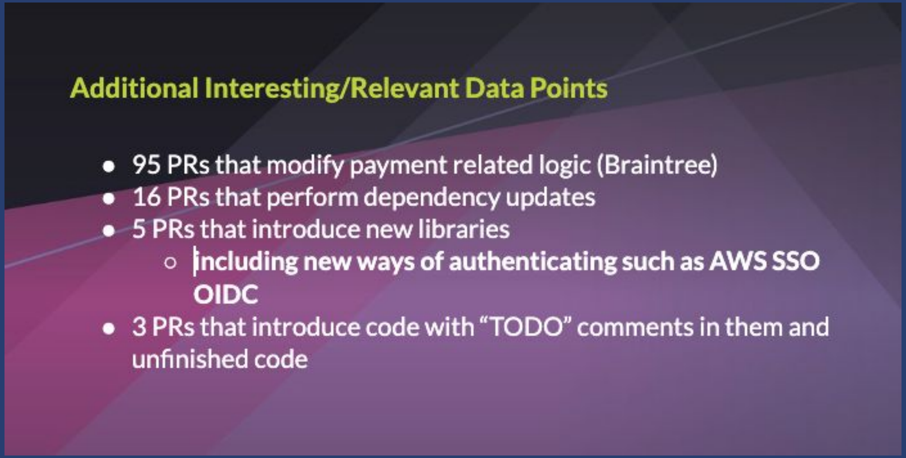
- `dojo/filters.py`: This file defines several Django filters that are used to filter and search for various objects in the Defect Dojo application. The key changes include:
  - Definition of various filter classes that inherit from the `DojoFilter` class, a custom `FilterSet` class that provides additional functionality for handling tags and other custom fields.
  - Filters cover a wide range of objects, including findings, tests, engagements, products, and endpoints, with each filter class having its own set of fields that can be used to filter the data.
  - Use of a variety of filter types, such as `CharFilter`, `NumberFilter`, `DateFilter`, `ChoiceFilter`, and `BooleanFilter`, to allow for flexible and comprehensive filtering.
  - Support for handling tags, including the ability to filter by tags, exclude tags, and search for tags that contain a specific pattern.
  - Implementation of custom methods, such as `custom_filter` and `custom_vulnerability_id_filter`, to provide additional functionality for filtering the data.
  - Support for ordering the results, with the ability to order by various fields.

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Innovations: Macro level changes

- Ability to summarize all changes and formulate / surface which are most impactful
- Idea came from performing risk assessments and realizing that we're able to aggregate all changes to determine which impact your security posture the most ->



Additional Interesting/Relevant Data Points

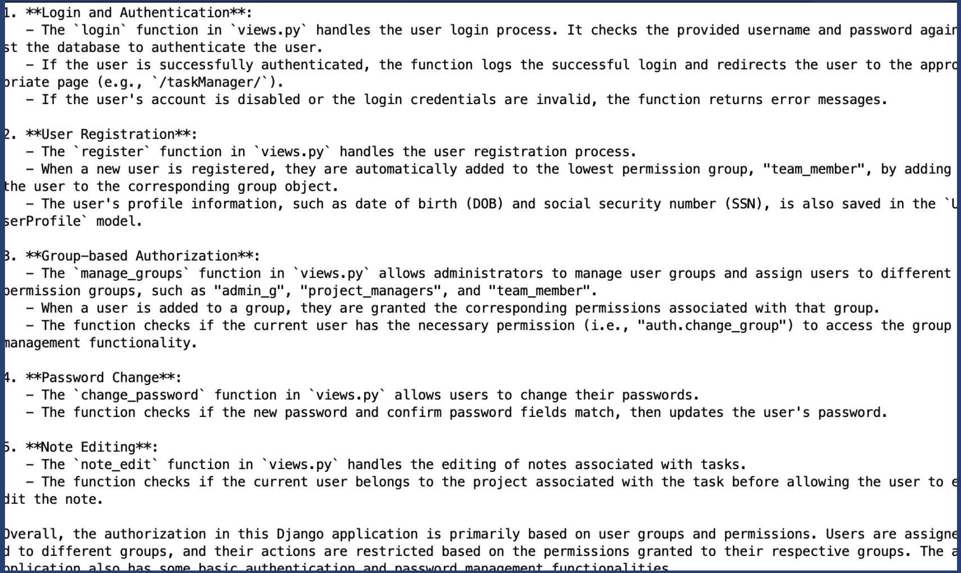
- 95 PRs that modify payment related logic (Braintree)
- 16 PRs that perform dependency updates
- 5 PRs that introduce new libraries
  - Including new ways of authenticating such as AWS SSO OIDC
- 3 PRs that introduce code with "TODO" comments in them and unfinished code

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Innovations: Compositional Analysis

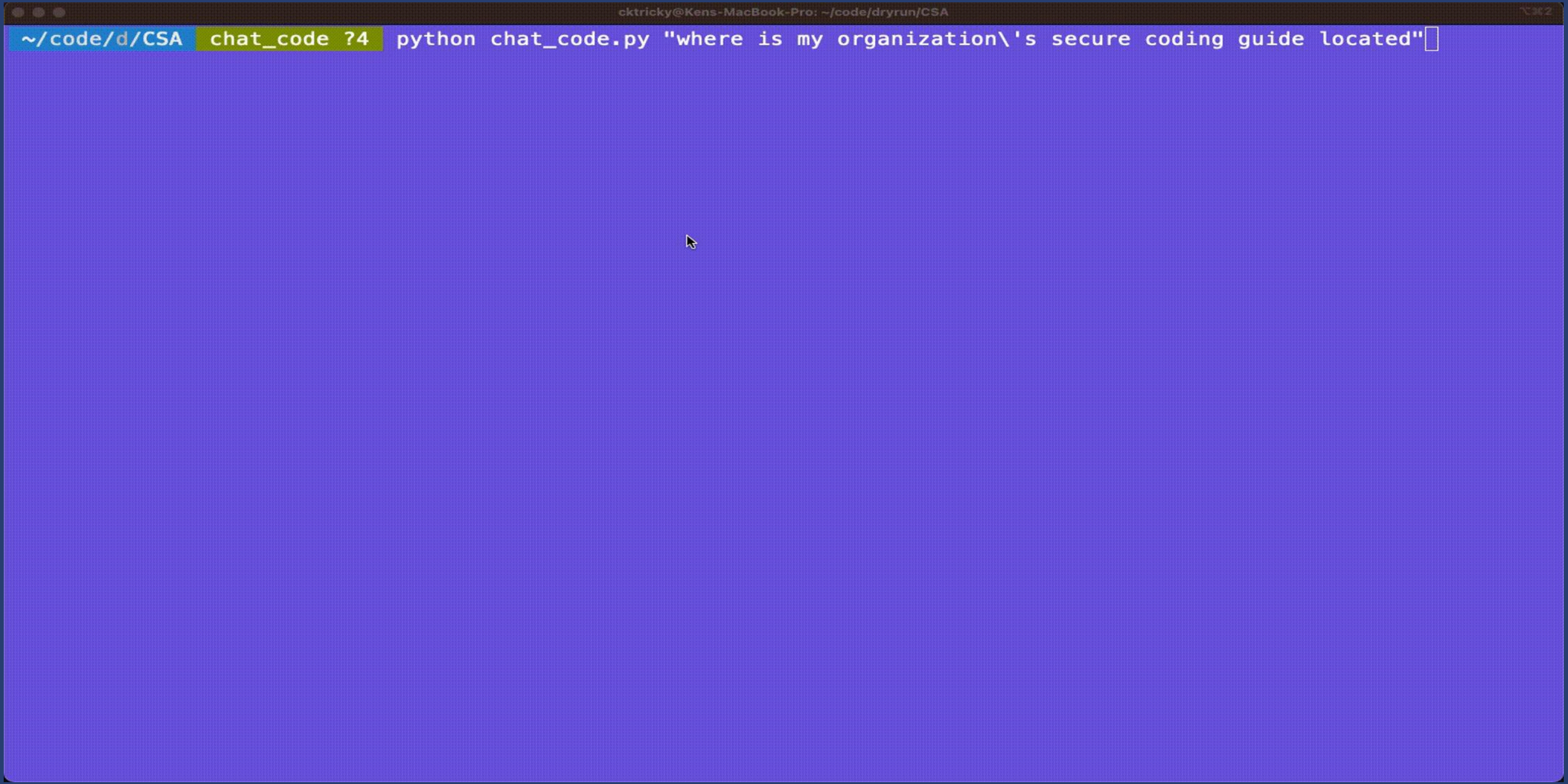
- Surface patterns specific to your application
- For example, one 🤯 moment for me was when I began experimenting with if AI could detect authn/z implementations and patterns… it can ->



```
1. **Login and Authentication**:
   - The `login` function in `views.py` handles the user login process. It checks the provided username and password agai
st the database to authenticate the user.
   - If the user is successfully authenticated, the function logs the successful login and redirects the user to the appro
priate page (e.g., `/taskManager/`).
   - If the user's account is disabled or the login credentials are invalid, the function returns error messages.

2. **User Registration**:
   - The `register` function in `views.py` handles the user registration process.
   - When a new user is registered, they are automatically added to the lowest permission group, "team_member", by adding
the user to the corresponding group object.
   - The user's profile information, such as date of birth (DOB) and social security number (SSN), is also saved in the `U
serProfile` model.

3. **Group-based Authorization**:
   - The `manage_groups` function in `views.py` allows administrators to manage user groups and assign users to different
permission groups, such as "admin_g", "project_managers", and "team_member".
   - When a user is added to a group, they are granted the corresponding permissions associated with that group.
   - The function checks if the current user has the necessary permission (i.e., "auth.change_group") to access the group
management functionality.

4. **Password Change**:
   - The `change_password` function in `views.py` allows users to change their passwords.
   - The function checks if the new password and confirm password fields match, then updates the user's password.

5. **Note Editing**:
   - The `note_edit` function in `views.py` handles the editing of notes associated with tasks.
   - The function checks if the current user belongs to the project associated with the task before allowing the user to e
dit the note.

Overall, the authorization in this Django application is primarily based on user groups and permissions. Users are assigne
d to different groups, and their actions are restricted based on the permissions granted to their respective groups. The a
pplication also has some basic authentication and password management functionalities.
```

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series



```
cktricky@Kens-MacBook-Pro: ~/code/dryrun/CSA
~/code/d/CSA  chat_code ?4  python chat_code.py "where is my organization\'s secure coding guide located"
```

DRYRUN
.SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Innovations

Opportunities to change our approach / roles

- Role change as defenders
  - Overseer, expert, last line triage
- Role change as consultants
  - Engineer, Process, AI
- Builders, focus more on the prevention than bugs
- Feed material to AI and work towards normalizing outputs

DRYRUN .SECURITY

Conclusion

DRYRUN
.SECURITY

Conclusion

- 💡 We've spent the last 26 years doing relatively the same things
- 💡 We have new opportunities to change our approaches
- 💡 Learn how LLMs and their ecosystems work
- 💡 **"Shift smart"**

DRYRUN .SECURITY

https://github.com/cktricky/speaking/tree/csp-summer-series

Contact Info

@cktricky

ken@dryrun.security

youtube.com/@AbsoluteAppSec

DRYRUN .SECURITY