Welcome to *API Vulnerability Detection: Automating Reconnaissance for More Efficient Penetration Testing.* This presentation is meant to serve as a proposal for research to fulfill the dissertation requirement for the MSc Cyber Security at the University of Essex. The content of this presentation will include: why APIs are the focus of this research proposal; Vulnerability detection current practice; There will be a research question in the form of a hypothesis which will then lead to a supporting literature review and relevant ethical considerations; A proposal for research will then be presented along with an estimated timeline; And we'll wrap everything up with some concluding statements.

APIs, or Application Programming Interfaces, are the focus of this research due to their comprehensive use in Industry 4.0. Major corporations utilise APIs to increase their customer base, for innovation and improved production, improved web integration, and flexibility for future needs. Governments worldwide, the banking industry, and healthcare are also heavily reliant on API services for these same incentives. APIs are effective because they can "expose system capabilities and facilitate machine-to-machine interfaces" (Bogle et al., 2022: 377) through endpoints which utilise the HTTP protocol.

But for all the benefits APIs bring, there are challenges to their security. APIs are subject to serious vulnerability concerns, and can be especially vulnerable to rate limiting, business logic, and broken function level authorisation vulnerabilities. Web application security and API security are unique from each other, differing most significantly in the depth and scale of attack. Endpoint exposure and exploitation can lead to microservice exposure, or data sharing exposure, the compromise of which would essentially allow an attacker full access to the database or service structure of a web application. Instances of API breach within Cloud computing is an example of this.

Additionally, traditional web application IDS detection methodology is often ineffective for API IDS detection, as the design-specific attack surface and customized architecture typical of APIs lead to more zero-day logic flaws or other complex attacks. Vulnerability detection for APIs must have a different set of requirements to be comprehensive, and this requirement can leave them open to exploitation. If aspects of API IDS are automated to run alongside traditional web app IDS, this may incentivise the incorporation of API-specific vulnerability testing without doubling the resources required.

One aspect essential to both web app and API IDS is reconnaissance. Without robust recon the reliability of any subsequent vulnerability test is diminished. But this is often the least enjoyable aspect of vulnerability testing as it is time intensive. A pentester or ethical hacker must manually search for and compile multiple pieces of information concerning a target. The breadth and depth of this data is a contributing factor in this regard, as there are multiple sources to scour and multiple data types to download or save.

Research into API vulnerability testing tends to focus on an aspect of endpoint detection such as fuzzing, on an area of utilisation such as the Internet of Things, or on a category of vulnerability such as error scenarios. Web application-specific literature has more examples of phase-based research concerning reconnaissance, exploit detection, and vulnerability exploitation. What this study endeavors to do is bridge the gap between API-specific vulnerability detection and automated reconnaissance. Much of this goal takes inspiration from Akshay et al.'s 2022 study focusing on comprehensive web application recon. Of particular interest was the researchers ability to produce an artifact utilising web crawling to identify subdomains and credentials with a multi-threaded technique which shortened the overall time requirement involved, and produced output which could be fed to other tools for further extrapolation.

This study would like to investigate the API-specific equivalent, which leads to the hypothesis that web crawling can automate key aspects of API reconnaissance to improve labour and time requirements for API vulnerability testing

The aims and objectives of this study would be to present an artifact demonstrating robust automated reconnaissance through the identification of aspects most adept for automation. Automation should provide essential information through program algorithms that identify, crawl, and categorise relevant data. This data should then be sorted and saved as output in relevant file formats for further investigation or manipulation depending on the needs of the test.

API-specific reconnaissance is focused around endpoints, which are the gateways to the API attack surface. Two stages made up API reconnaissance: passive recon and active recon. This study is interested in passive recon, which focuses on locating API endpoints, credentials, and documentation, all of which are particularly applicable to web crawling. Within passive recon there are three stages. Again, this study is interested in only one phase, phase one of the passive recon process, again due to web crawling compatibility.

Phase one focuses on the breadth and depth of a target's open source resources online. OWASP Amass, google 'dorking' (using the google search engine with primed search terms), Shodan, and ProgrammableWeb are common resources for API pentesters to gather large swaths of information. Amass is a third party application, and so will not be subject to web crawling, but is included in the overall automation arti fact as it provides a "full map of [the] digital asset" (Bhavsar & Chudasama, 2021: 3) including the attack surface, DNS enumeration, and asset location. Omitting this data from automation would result in an incomplete API profile. The remaining resources would be best utilised through focused web scraping and web crawling.

Web scraping is use of the HTTP protocol for data extraction on a single target, which is particularly relevant for API subdomain data, much of which uses the HTTP protocol. Web scraping can be key-word specific, thus limiting any unnecessary breadth in a target search. Locating data for download can use vertical sampling, which explores the depth of a target before moving forward, or horizontal sampling, which explores the breath of target before focusing more deeply. Web crawling can be considered a scaled extension of web scraping, where multiple web targets are scraped for relevant information.

This is performed by link hopping and by web crawler algorithms utilising a robot.txt document if applicable. Web crawling can span through both the surface and dark web, and can utilise the TOR channel for crawling depth, though the surface web would more likely use only crawling depth for intensive scraping. Like horizontal and vertical data sampling, there are general and focused crawlers, the use of either depending on the needs of the program. Focused crawlers are perhaps best suited for API reconnaissance as they discriminate between relevant and irrelevant domains.

Hossain et al. present an example focused crawler, simple in its design but comprehensive in its crawling abilities. This crawler is based on the decision tree model wherein URLs are extracted from a queue and parsed vertically while relevent information is saved for future crawling.

The decision tree cannot terminate until certain conditions are met, thus allowing the crawling depth to expand as necessary. But it should be noted this could lead to server overloading not unike a Denial of Service attack. Relevant information is determined with a fitness value; in this example researchers utilised Naive Bayes binary classification, but multivariate keyword weighting algorithms are also possible. In addition, the crawler output can be validated by the p-Value statistical equation.

The programming language perhaps best suited to build a crawler such as this is Python, as this language has been proven suitable for web scraping due to it’s popularity and syntax. The Python community is large and willing to help if programmers are in need. It is also suitable for ethical hacking, as it can handle object oriented programming and has extensive open source libraries for almost any need. Python is also suitable for automation, but it should be noted that as Python is a high-level coded language it is difficult to secure. Measures should be taken to mitigate vulnerabilities in any automated program.

Ethical considerations are also incredibility important for automation. A vulnerability test can turn into an attack if not diligent. Pentesters and ethical hackers must not extend beyond agreed upon attack surfaces, nor should they target unwilling web applications for reconnaissance. Web crawler design must respect the bounds of the web app’s crawler policy, whether through link hopping or robot.txt. The scraping rate should mimic a single person’s activity and should not overburden the server, which could cause a failure of service akin to a DOS attack. And always, privacy of individuals, including credentials or sensitive documents, must comply with GDPR regulations. Unintended use of the artifact is also a concern. Though this study is meant to aid pentesters and ethical hackers, malicious actors would also have access to any information provided. Precautions should be taken to limit malicious use of the study.

With this in mind, this study intends to automate passive reconnaissance for API vulnerability testing using a python framework. OWASP Amass and web crawling will comprise the automated modules which will locate subdomains, endpoints, credentials and documents. These data will be saved in .pdf and .csv formats for further manipulation as necessary.

Intial recon would engage the target in subdomain enumeration with Amass and web crawling and scraping with customised python modules. This would require multi-threading for time efficiency. The web crawler would be a focused decision tree model set to mimic the crawling abilities of a single person. Vertical sampling would be employed to locate endpoints, credentials, and documents, while a Naive Bayes classifier would assign a fitness value. This step could also possibly include a secondary weighted keyword analysis for data sorting. Relevant data are exported as output along with the Amass enumeration data to be downloaded as either PDF or CSV files. p-Value statistical analysis would have to be carried out on the output to assess crawler reliability.

The study would also seek to address ethical considerations by engaging only willing targets through gaining permission. Web crawlers would have parameterised limited scrape rates and crawls rates, and parameters would also be set to respect robot.txt files and avoid sensitive information disclosure. Malicious actors may be thwarted by reproducibility; the study intends to include enough information to be reproducible without disclosing key aspects which could entice malicious actors.

This study would follow the timeline proposed by the university: the first month would comprise the research outline and beginning proposal development. The literature survey and project development and write up would comprise weeks 5 to 28, while artifact preparation and presentation would comprise weeks 20 to 30. This should give enough time for the artifact to be completed.

In conclusion, this presentation has discussed the importance of APIs, the importance of reconnaissance, web crawling utilisation for automated reconnaissance and ethical concerns therein, presented a research question and proposal, and discussed a timeline for project completion. Thank you for your kind attention.

References

Ahmed, R. et al. (2022) Machine Learning and Deep Learning Approaches for CyberSecurity: A Review. *IEEE Access*, 10: 19572 – 19585

Akshay S et al. (2022) Automation of Recon Rocess for Ethical Hackers. In: *2022 International Conference for Advancement in Technology, Goa, India, 21 – 22 January, 2022.* IEEE: 1 – 6

Amale et al. (2021) SpyDark: Surface and Dark Web Crawler. *2021 Second International conference on Secure Cyber Computer and Communication*. IEEE: 45 - 49

Arauza, O., Brewer, R., Hart, T., & Westlake, B. (2021) The Ethics of Web Crawling and Web Scraping in Cybercrime Research: Navigating Issues of Consent, Privacy, and Other Potential Harms Associated with Automated Data Collection. In: Holt, T. J. & Lavorgna, A. (eds.) *Researching Cybercrimes.* Switzerland. Springer: 435 - 456

Ariffin, M. A. M., Ibrahim, M. F., & Kasiran, Z. (2020) API Vulnerabilities in Cloud Computing Platform: Attack and Detection. *International Journal of Engineering Trends and Technology*: 8 – 14

Arnold, T & Seitz, J. (2021) Blackhat Python: Python Programming for Hackers and Pentesters. 2nd Ed. San Francisco, USA: No Starch Press.

Arun, A. et al. (2022) An Automated Word Embedding with Parameter Tunde Model for Web Crawling. *Intelligent Automation & Soft Computing*, 32 (3): 1617 - 1632

Badhwar, R. (2021) Intro to API Security – Issues and Some Solutions! In: *The CISO's Next Frontier*. Cham, CH. Springer: 239 – 244

Ball, C. J. (2022) Hacking APIs: Breaking Web Application Programming Interfaces. San Francisco, CA, USA. No Starch Press.

Begum, A., Bhuiyan, T., Hadid, I., & Rahman, S. (2018) API Vulnerabilities: Current Status and Dependencies. *International Journal of Engineering & Technology*, 7: 9 -13

Biswas, P. & Nigam, H. (2021) From Web Scraping to Web Crawling. In: A. Choudhary et al. (eds.) *Applications of Artificial Intelligence and Machine Learning*. Singapore. Springer: 97 - 112

Bhavsar, R. & Chudasama, D. (2021) Technical Methods of Information Gathering. *Journal of Web Engineering & Technology*, 8 (3): 1 - 5

Bogle, A., Mahmood, R, Pennington, J., Tran, T., & Tsang, D. (2022) A Framework for Automated API Fuzzing at Enterprise Scale. In: *IEEE Conference on Software Testing, Verification and Validation*. IEEE: 377 – 388

Bouchard, M., Frank, R., & Westlake, B. (2017) Assessing the Validity of Automated Webcrawlers as Data Collection Tools to Investigate Online Child Sexual Exploitation. *Sexual Abuse*, 29 (7): 685 - 708

Ceccato, M. et al. (2022) Automated Black-box Testing of Nominal and Error Scenarios in RESTFUL APIs. *Software Testing, Verification and Reliability*, 32 (5): 433 – 442

Díaz-Rojas, J. A., Limón, X., Ocharán-Hernández, J. O., & Pérez-Arriaga, J. C. (2021) Web API Security vulnerabilities and Mitigation Mechanisms: A Systematic Mapping Study. In: *9th International Conference in Software Engineering Research and Innovation*. IEEE: 207 – 218

Dong, L et al. (2021) IoT-APIScanner: Detecting API Unauthorized Access Vulnerabilities of IoT Platform. *IEEE Internet of Thing Journal*, 8 (13): 10327 - 10335

Feng, H., Fu, X., Sun, H., Wang, H., Zhang, Y. (2020) Efficient Vulnerability Detection Based on Abstract Syntax Tree and Deep Learning. In: *IEEE Conference on Computer Communications Workshops, Toronto, ON, CA*. IEEE: 722 - 727

GDPR (2018) General Data Protection Regulation (GDPR). General Data Protection Regulation (GDPR). [Available Online]: https://gdpr-info.eu/

Gu, G. & Mendoza, A. (2018) Mobile Application Web API Reconnaissance: Web-to-Mobile Inconsistencies & Vulnerabilities. In: *2018 IEEE Symposium on Security and Privacy*. IEEE: 756 – 769

Gupta, A., Singh, K. B., & Singh, R. K. (2018) Web Crawling Techniques and It's Implications. *Globus: An International Journal of Management & IT*, 9 (2): 1 - 7

HackerOne (2023) *Hacktivity | HackerOne.* hackerone.com [Available Online] https://hackerone.com/hacktivity/overview

Hossain, S. A., Nobel, N. I., Rahman, A. K. M. S., Shamrat, F. M. J. M., Tasnim, Z. (2020) An Effective Implementation of Web Crawling Technology to Retrieve Data from the World Wide Web (WWW). *International Journal of Scientific & Technological Research*, 9 (1): 1252 - 1256

Huang, G., Li, J., Ren, J., Zhang, B. (2021) Efficiency and Effectiveness of Web Application Vulnerability Detection Approaches: A Review. *ACM Computer Survey*, 54 (9): 1 – 35

Irfan, MD et al. (2023) API Traffic Anomaly Detection in Microservice Architecture. In: *IEEE/ACM 23rd Symposium on Cluster, Cloud, and Internet Computing Workshops.* IEEE: 206 – 213

Khder, M. A. (2021) Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. *International Journal of Advances in Soft Computing & Application*, 13 (3): 144 - 168

Li, V. (2021) Bug Bounty Bootcamp: The Guide to Finding and Reporting Web Vulnerabilities. San Francisco, USA: No Starch Press.

Munsch, A. & Munsch, P. (2021) The Future of API (Application Programming Interface) Security: The Adoption of APIs for Digital Communications and the Implications of Cyber Security Vulnerabilities. *Journal of International Technology and Information Management*, 29 (3): 25 – 45

Navaneethan, C. & Rajiv, S. (2021) Keyword Weight Optimization Using Gradient Strategies in Event Focused Web Crawling. *Patter Recognition Letters*, 142: 3 - 10

Ntantogian, C., Stasinopoulos, A., & Xenakis, C. (2018) Commix: Automating Evaluation and Exploitation of Command Injection Vulnerabilities in Web Applications. *International Journal of Information Security*, 18: 49 – 72

OFFSEC (2023) *OFFSEC's Exploit Database Archive | Exploit Database.* exploit-db.com. [Available online] https://www.exploit-db/com/

Paxton-Fear, K. (2022) *API Hacking Toolbox w/ Dr. Katie Paxton-Fear | Traceable AI*. youtube.com. [Available online] https://www.youtube.com/watch?v=qC8NQFwVOR0

Permual, A., et al. (2021) Cybercrime Issues in Smart Cities, Networks and Prevention Using Ethical Hacking. In: C. Chakraborty et al. (eds.) *Data-Driven Mining, Learning and Analytics for Secured Smart Cities.* Switzerland. Springer: 333 - 358

Siriwardena, P. (2020) Advanced API Security: OAuth 2.0 and Beyond. 2nd Ed. New York, NY, USA. Apress.

Shamunesh P, Srinivas, L. N. B., Vinoth S (2023) Cybercheck – OSINT & Web Vulnerability Scanner. In: *Second International Conference on Edge Computing and Applications.* IEEE: 275 – 279

University of Essex (2023) Computing Department – MSc Project Roadmap. [Available Online]: https://www.my-course.co.uk/course/view.php?id=10163

Images

Castro, A. (2020) *Amazon Logo* | *The Verge.* theverge.com [Available Online] https://www.theverge.com/2020/7/30/21348368/amazon-q2-2020-earnings-covid-19-coronavirus-jeff-bezos

CMM (n.d.) Under Lock and Key. Care Management Matters [Available Online] https://www.caremanagementmatters.co.uk/feature/under-lock-and-key-making-the-nhs-and-social-care-cyber-safe/

Green Imaging (n.d.) *Healthcare* | *Green Imaging.* greenimaging.net [Available Online] https://greenimaging.net/what-is-going-on-with-healthcare/

IBM (2023) *IBM Logo* | *Forbes*. forbes.com [Available Online] https://www.forbes.com/sites/moorinsights/2023/07/07/ibm-watsonx-empowers-businesses-to-build-tune-and-deploy-reliable-generative-ai-models/?sh=35383a022fda

Kazmierski (2019) *UN Flags* | *Shutterstock.* Ranking Digital Rights [Available Online] https://rankingdigitalrights.org/2019/09/26/what-should-governments-do/

LogicRays (2020) Python Logo | LogicRays Academy Blog. logicraysacademy.com [Available Online] https://www.logicraysacademy.com/blog/what-is-python-programming-language/

Meta (2023) *facebook Logo. f*acebook.com [Available Online] https://th-th.facebook.com/

Netflix (2023) *Netflix Logo*. netflix.com [Available Online] https://about.netflix.com/en/news/announcing-basic-with-ads-us

Newsom, N. (2010) A Red Graph on the Rise Over Stacks of Gold Coins. Alamy [Available Online] https://www.alamy.com/stock-photo-a-red-graph-on-the-rise-over-stacks-of-gold-coins-35005260.html?imageid=2669D11F-627A-4CE6-A542-2191DE571843&p=136117&pn=1&searchId=b63614e611cfdf67bf8919cedd2b1f54&searchtype=0

Planview (2023) *Salesforce Logo*. planview.com [Available Online] https://www.planview.com/products-solutions/products/hub/integrations/salesforce/

Rayburn, D. (2021) *API* | *Streaming Media Blog*. streamingmediablog.com.

ScrapingBot (2023) How to Build a Web Crawler. scraping-bot.io [Available Online] https://www.scraping-bot.io/how-to-build-a-web-crawler/

Sofi (2022) *Commercial Banking | SoFi Learn.* sofi.com [Available Online] https://www.sofi.com/learn/content/what-is-commercial-banking/

StickPNG (n.d.) Download Malicious Hacker Transpartent PNG. stickpng.com [Available Online] https://www.stickpng.com/img/icons-logos-emojis/emojis/malicious-hacker

ThreatPost (2018) Navigating an Uncharted Future, Bug Bounty Hunters Seek Safe Harbors. threatpost.com [Available Online] https://threatpost.com/navigating-an-uncharted-future-bug-bounty-hunters-seek-safe-harbors/133202/

Uber Technologies, Inc. (2023) *Uber Logo | Uber*. Google Play Store [Available Online] https://play.google.com/store/apps/details?id=com.ubercab&hl=th

Walgreens (2023) *Walgreens Logo | Newsroom.* Walgreens Newsroom [Available Online] https://news.walgreens.com/