

VulnTraq – A Free Open Source Vulnerability Ticket Tracker System

1. Introduction

Successful vulnerability management is part of maintaining organizations' security postures. The stakes could not be higher. Per a study conducted by the *Ponemon Institute* for *ServiceNow*, 60% of breaches in 2019 were linked to a vulnerability where a patch was available but not applied. This is caused by patches either not being applied at all or applied with a significant delay. The report indicates that the remediation process taking too long is a general rule rather than the exception. Specifically, vulnerabilities took an average of 12 days to patch. Critical ones took even longer – an average of 16! [1]

Unsuccessful vulnerability management thus significantly increases the risk of a successful breach. The risks to an organization of such *can not be overstated*; they cost an average of \$4.24 million in 2021 per a joint *IBM* and *Ponemon Institute* report. [2] Per the *National Cybersecurity Alliance*, Small & Medium-sized Businesses (SMBs) are especially more susceptible to a breach's consequences because [3]:

- 62% of all cyberattacks are directed against them as they are an easier target for attackers than larger enterprises
- Average small businesses needed \$690,000 to recover
- Middle-sized companies needed to spend more than \$1,000,000 to recover

It is unfortunate that SMBs facing such risks are severely constrained in their ability to harden their security postures. According to Kaspersky's 2021 *IT Security Economics* report, their average budget was *only* \$267,000 on average. [4] The fact that SMBs have limited security budgets, are disproportionately targeted, and that a successful breach can cause a sizable blow to their financial and operational health makes robust and inexpensive vulnerability management a top priority.

2. Current State of Vulnerability Management

The vulnerability management process in SMBs generally involves the following steps:

1. **Gathering Initial Information:** Information regarding the vulnerable system(s), severity, and whether a patch is available and exploitation is ongoing is gathered from news articles, threat intelligence feeds, and trusted advisory sources
2. **Triage and Risk Assessment:** The security team inspects the organization's environment to determine whether it is affected. If the vulnerability is confirmed to be present, the team gauges its extent in the environment, ease of exploitation, whether mitigationary controls are available, the current risk it poses, and whether that risk would be currently considered acceptable for the organization's needs. The process terminates here if the current level of risk is considered acceptable.

3. **Opening Patching Ticket for IT Staff:** If the risk that the vulnerability poses to the organization is not considered acceptable, the security team opens a patching ticket for the IT staff to complete. The ticket specifies the course of remediation to undertake and the timeframe by which it is to be completed
4. **Monitoring Patching Status and Whether the Ticket Has Been Closed:** The security team periodically reviews all previously-opened patching tickets to determine whether they have been closed by the specified deadlines. If a ticket has not been closed by the time of the deadline, the security team will take appropriate remediation steps

This process is relatively simple and straightforward. However, the process of tracking patching ticket statuses and whether they have been closed by the specified deadlines can become more complex than may be initially desired.

2.1 Current State of Vulnerability Management – Tracking Patching Tickets

Vulnerability patching tickets can be tracked within a traditional enterprise-wide ticketing system. Staff can either manually search for previously-created tickets to inspect their information or manually develop custom tracking workflows and graphs to do this on their behalf. However, both options are problematic as

- Manually searching for each individual patching ticket takes extensive time and effort with the risk of staff accidentally forgetting to search for a particular ticket
- Patching-ticket tracking workflows and graphs need to be custom-made. The process of developing and testing such capabilities on a per-SMB basis involves extensive and redundant labor costs

Excel spreadsheets are a simpler and commonly-used alternative. They are usually stored in an enterprise's central file share server and opened by users on their workstations. The information tracked includes, but is not limited to, the vulnerability name, patching ticket status [whether a ticket is currently opened or closed], patching ticket ids, the dates tickets were opened and closed.

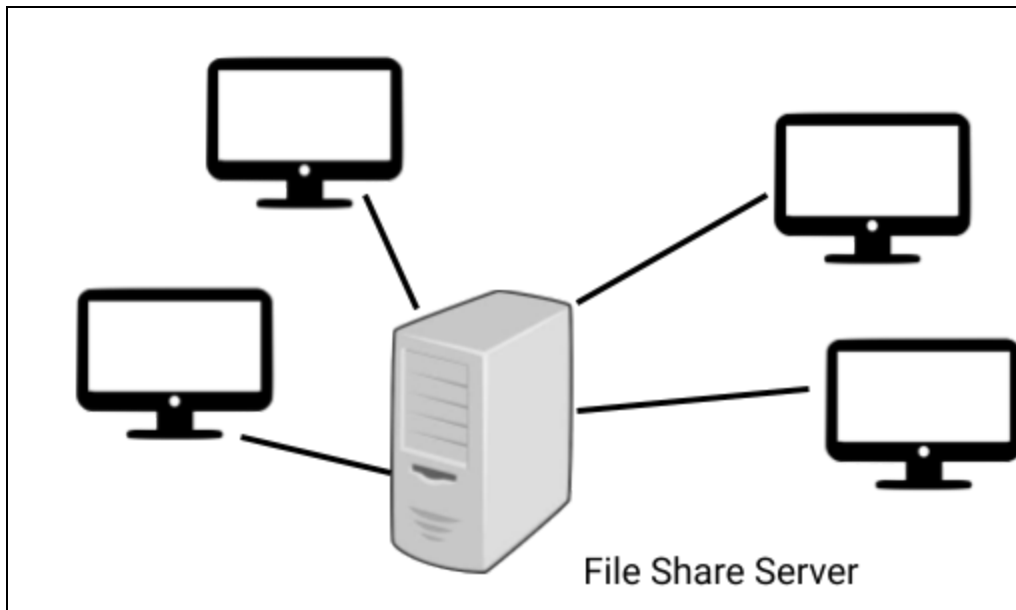


Figure 1: Conceptual Diagram of an Enterprise Using Excel Spreadsheets for Vulnerability Tracking

Such an Excel-based approach has significant inefficiencies. First, the process of opening the Excel spreadsheet from the file share server in end users' Excel applications and saving changes to it can be a slow process. Moreover, there is no template for Excel spreadsheet trackers with data graphing functionalities that is considered to be a definitive industry standard. Both SMBs and larger organizations typically create a brand new template with desired capabilities when starting their vulnerability management programs. For SMBs, the time and resources invested in developing such a spreadsheet and refining its tracking and graph-generation capabilities can be significant and better spent elsewhere. Finally, a spreadsheet being stored and available from the single file share server poses potential operational issues in terms of ensuring changes are properly synchronized and file versions do not clash. Synchronization is a known problem, addressed by cloud-based tools like Google Drive and Microsoft OneDrive. However, organizations are reluctant to adopt such tools and entrust them with sensitive vulnerability-handling data, with 90% very/moderately concerned with public cloud security per Bitglass' 2015 *Cloud Security Report*. [5]

2.2 Current State of Vulnerability Management – Available Vulnerability Handling Tools

There are multiple vulnerability handling tools available, both open source and commercial. This list was compiled after extensive research:

- **Open Source:** These primarily consist of vulnerability scanners such as OpenVAS, nmap, Vulners, and vuls.io. They identify vulnerable systems but do not track opened and ongoing patching tickets specifically
- **Commercial:** While there is a multitude, the discussion will only discuss two for the sake of brevity: *Alienvault's USM Anywhere Vulnerability Management & SIEM* and *ManageEngine Vulnerability Manager Plus*.

Alienvault's vulnerability manager collects data from SIEM/log management tools, vulnerability scanners, and threat intelligence feeds to streamline patch management and threat response. However, its price is exorbitant for SMBs' budgets: anywhere from £900 to £85,000 a year. [6]

The *ManageEngine* vulnerability manager is a robust solution for managing all aspects of the vulnerability management lifecycle. Supported capabilities include but are not limited to: centralized patch management, testing and deployment, automated vulnerability management and detection, and verifying compliance with *CIS Benchmarks & Compliance* standards. [7] While it is free for small businesses up to 25 devices, its *Professional* and *Enterprise* versions can respectively cost \$695 and \$1195 per year for each group of 100 workstations, not to mention additional add-on capabilities. [8] Depending on business size, revenue, and balance sheet, such an expenditure may not be affordable.

Hence, there is not a good vulnerability patching-ticket management solution available that both meets SMBs' operational needs and budgetary constraints.

3. Problem To Be Solved

In light of the current state of vulnerability management using Excel to track ongoing vulnerability remediation processes and tickets and the lack of a good tracking solution that both meets SMBs operational needs and budgetary constraints, a new solution must be developed.

It must be free and open source to address even the smallest of SMBs' budgetary constraints. Additionally, it must avoid the pitfalls of the popularly-used Excel spreadsheet approach by supporting multiple users, working "out of the box" without excessive administrator set up efforts, generating useful reports automatically, and working in conjunction with a free & open source enterprise ticketing system that SMBs are likely to use.

4. Solution

A prototype solution called *VulnTraQ* was developed during the Summer 2022 semester to address the problem facing SMBs in performing vulnerability management. Its conceptual diagram is as follows:

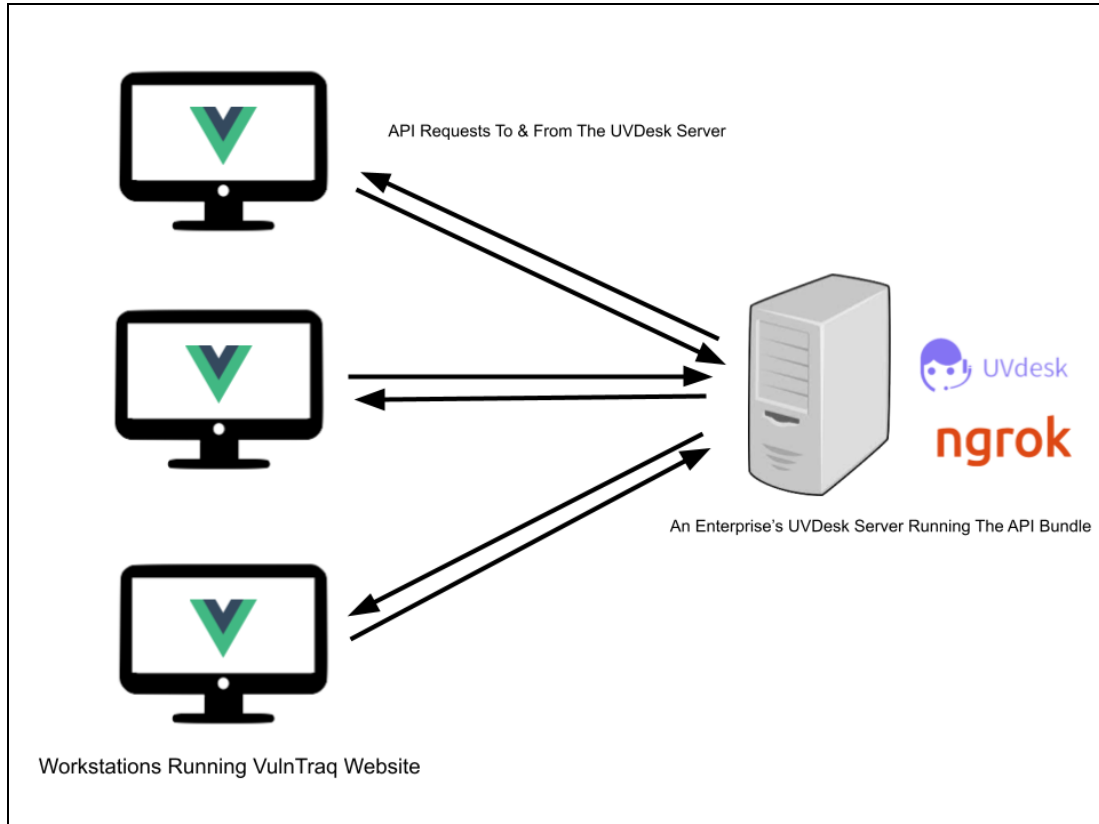


Figure 2: Conceptual Diagram of the VulnTraq Prototype Deployed In An Operational Environment

The application consists of two portions: the backend and the frontend.

4.1 Solution – The Backend

The backend is designed to be a server deployed in an enterprise's local network. It should be noted that it is irrelevant whether it is virtualized or installed to run directly on server hardware.

The server runs the latest version of Ubuntu with an application called UVDesk executing at the <http://localhost:8080> URL through an installed Apache server. UVDesk is a free and open-source enterprise ticketing application typical of those used by SMBs. In the setting of an enterprise using *VulnTraq*, all IT administrators would have direct access to the UVDesk application to inspect, create, and close tickets. It should be noted that in the implementation of this prototype, the server was created as a locally-executing VirtualBox Ubuntu 20.04 Virtual Machine with the UVDesk installed per the 2021 instructions specified on the *Computing for Geeks* website. [9] While neither the backend nor its frontend integrates with any outside systems for keeping track of issues or vulnerabilities, UVDesk's ticket-handling workflow capabilities gives administrators the opportunity to customize the ticket-handling process for optimal handling efficiency.

Because it does not come with API integrations by default, its API bundle extension was installed. [10] It provides the functionality for creating a key for external clients to use and the

API commands necessary to query the server to access and perform operations on tickets' data.

The following settings **must** be configured in the backend's UVDesk application when it is being set up:

- **Agents:** They are an enterprise's IT staff. All must have a corresponding profile created and be assigned to the *Group(s)* and *Team(s)* corresponding to the system(s) they support
- **Groups and Teams:** They are groups of an enterprise's IT staff categorized by the system(s) they support. In this specific prototype, they were: Browsers, Databases, VMs, and Exchange
- **Ticket Types:** These are the possible types of vulnerability remediation tickets that can be created, named by their criticality. In this specific prototype, they were:
 - URGENT PATCH – NOW
 - HIGH PATCH – 10 DAYS
 - MEDIUM PATCH – 30 DAYS
 - LOW PATCH – 3 MONTHS

For *VulnTraq* to properly interpret the specified criticalities and calculate the due dates for ticket closure, the names of the ticket types must either contain the word "Now" or a number followed by the words "DAY" or "MONTH".

An application executing locally within the context of the server is inaccessible to outside clients by definition. A reverse proxy application was thus necessary to forward communications to and from the outside clients to the locally-executing UVDesk application. Because repeated efforts at repurposing the *Apache Server* running UVDesk to include reverse proxying capabilities and configuring *nginx* for the task proved unsuccessful and time was in short supply, *ngrok* was selected as the reverse proxy of choice. *Ngrok* is a secure tunneling service which allows for a locally-executed application to be publicly accessible with minimal effort. The developed prototype used its free version.

4.2 Solution – The Frontend

The frontend is designed to collect information regarding all tickets stored in the UVDesk backend, process it, generate an informational table reports, and display it in a convenient fashion for the end user to review and work with. It is built as a Vue.JS website and uses the following JavaScript frameworks:

- **Axios:** An open-source library for making HTTP requests to endpoints. [11] *VulnTraq's* frontend uses it to make HTTP and REST API requests to the backend server
- **Bootstrap Vue:** A library for the rapid development and usage of graphical components in a Vue.JS frontend

- **Vue-ChartJS:** A framework for quickly developing robust and sleek-looking graphs for display in Vue.JS websites. *VulnTraq's* frontend uses it to generate and display the graph of the number of tickets opened and closed each day

Upon loading, the frontend website queries the backend server to collect information regarding all previously-created tickets and stores the retrieved data in *Vuex*, the global state manager for Vue.JS applications. The data is then processed and graphically presented for the user to read and work with. Users can select individual tickets presented in the informational table graph, open a panel displaying generated graphs, or create new tickets.

With regards to the ticket-creation process, this prototype only supports the manual creation of new patching tickets. Users specify the ticket's subject line, message, patching priority level [urgent/high/medium/low], and the system-type patching group to which it is to be assigned [e.g. browsers, exchange, servers, databases, etc]. Lastly, ticket-creators upload a .CSV file listing the affected systems with any additional comments they may see fit to add.

It should be noted that in the implementation of this prototype, the frontend was locally running in the user's workstation. In a real enterprise setting, the frontend would be hosted on a server and/or hosting service like Digital Ocean. Such would allow for multiple users to view the website simultaneously and prevent individual workstations from becoming inundated with calculations processing the data received from the backend.

The frontend's code is available in the <https://github.com/secnate/VulnTraq> Github repository. It should be noted that its *main.js* file is home to the lines configuring the information needed for successful communication with the UVDesk backend through its API: its URL and API key.

Specifics regarding the frontend's looks and capabilities can be found in *Appendix A*.

4.3 Solution – Deployment Requirements

The specific requirements for deploying this prototype in an environment are minimal and are as follows:

- An on-premise physical or virtualized backend server running the UVDesk application and the *ngrok* reverse proxy. Its operating system is Ubuntu
- An on-premise or cloud-based server or service [e.g. Digital Ocean] hosting the Vue.JS frontend website
- The enterprise's DNS server having A Records mapping (1) the backend server's IP address to the *ngrok*-generated URL and (2) the frontend website's domain to the hosting web server's IP address

There are no additional *specific* requirements to the aforementioned in terms of servers, hardware, nor operating system requirements. Each deployment may be affected by the particulars of the environment into which it is deployed.

5. Prototype Limitations

The developed prototype has several categories of limitations related both to its architecture and operational and computational efficiency.

5.1 Prototype Limitations – UVDesk API

The prototype was developed with the Vue.JS frontend web application querying the backend UVDesk server through its API to read and manipulate ticket-related data. The installed UVDesk application did not come installed with API support by default. Hence, a special extension called the *API Bundle* had to be installed from its Github page. [10]

Usually API documentation for any service comes with great detail and even a *SwaggerHub* API design and documentation platform page for testing out commands. However, the API Bundle's documentation was a single tersely written Github wiki page. [12] Furthermore, the UVDesk website's extensive API documentation and linked *SwaggerHub* page were not applicable because they were for the **prior and outdated** API version. [13] The full extent of listed API commands' functionalities had to be inferred from repeated API request simulations with *Postman*. If the provided documentation was not as sparse, more time would have been available for developing and refining additional report-generation capabilities.

Additional challenges with provided API capabilities resulted in the frontend performing extra inefficient operations. For example, a **POST /api/v1/ticket** command needed to be invoked to create a new ticket when the ticket-creation modal's submit button was clicked. However, the command did not support specifying the ticket's patching priority level nor the patching group to which it is to be assigned. Invoking the **PATCH /api/v1/ticket/{ticketId}** command on a newly-opened ticket to specify its patching priority level and the patching group to which it is to be assigned was considered to be the best option. Repeated tests of the **PATCH /api/v1/ticket/{ticketId}** API command using *Postman* were unsuccessful. The lack of sufficient and robust capability documentation made the problem of reverse-engineering the settings necessary for the PATCH command to succeed even more difficult. Because the project was developed within tight time constraints with no extra time to spare on reverse engineering API commands, a creative workaround was devised. Specifically, a to-be-created ticket's patching priority level and patching group were *appended* to the text of its message before ticket data was submitted for publication. When receiving information for previously-created tickets, the then frontend had to perform extra operations to parse out the appended data from the ticket's original message. While such extra computations for a single ticket are barely noticeable, they compound over the course of many. The frontend performing such calculations means that the prototype is not functional with the desired level of operational efficiency.

It should be noted that the issue of the frontend being inefficient with extra operations performed to work around API-related issues *can be resolved*. Specifically, the UVDesk backend application *can* be modified to implement custom API commands. [14] However, such requires extensive time and resources to fully configure and test. The newly-configured UVDesk

application would then need to be repackaged and deployed to a Github repository for others to download and use. Such a course of action was not pursued due to tight time constraints.

5.2 Prototype Limitations – Architecture + Processing

The developed and deployed Vue.JS website frontend and the backend UVDesk server were not deployed onto any hosting service. For example, the Vue.JS frontend could have been deployed into a *Digital Ocean* hosting service and the Ubuntu VM running UVDesk is deployed into Google Cloud. The extensive task of service deployment did not occur over the course of this semester due to strict time and financial constraints. Instead, both the Vue.JS frontend and the VirtualBox Virtual Machine running UVDesk ran locally within the scope of a single computer.

It should be noted that the computer used for developing and testing the prototype has extra memory installed due to it being originally built for heavy-duty video processing. Even then, there has been occasional slowness when running both simultaneously. These operational constraints meant that *only one* frontend instance could be tested for its interaction with the backend. This is an especially significant limitation given that enterprises using *VulnTraq* would usually have more than one user working with it. Further testing would need to be conducted to evaluate how multiple simultaneously executing frontend instances communicating with the UVDesk server would affect it and each other.

A minor inconvenience in the *VulnTraq* prototype is the consequence of using the *ngrok* proxy's free version for making the UVDesk application running within the server's local context externally accessible. There were no issues with the reverse proxy's functionality itself. However, *ngrok's* free version creates a randomly generated URL with no intrinsic meaning. That is, instead of having a url such as *vulntraqbackend.ngrok.io*, the URL can have a value of <https://7574-75-183-154-210.ngrok.io>. Using a paid *ngrok* version can resolve that issue.

5.3 Prototype Limitations – Evaluation

Evaluating *VulnTraq's* capabilities and whether they improve vulnerability management programs' operational efficiencies requires extensive field-testing in at least one operational environment. The length of time over which it is to be tested is inversely correlated to the number of vulnerabilities an enterprise receives on a monthly basis. This is due to the fact that the more often an enterprise handles vulnerabilities, the less time it takes to collect the necessary data.

Such an evaluation would use objective numerical metrics collected from both before and during *VulnTraq's* operational usage over identical time periods. These include, but are not limited to:

- **Average Time To Patch:** The average time elapsed since a patching ticket is opened until it is closed
- **Percentage of Tickets Closed By Patching Deadline:** The percentage of all vulnerability-handling tickets closed before the patching deadline

- **Percentage of Tickets Open Past Patching Deadline:** The percentage of all vulnerability-handling tickets *not* closed by the patching deadline
- **Average Number of Minutes** vulnerability handlers spend on vulnerability-related duties daily

Such an extensive field test in at least one enterprise that collects metrics from both before and after *VulnTraq*'s deployment was not possible over the course of the compressed Summer 2022 semester. Another evaluation method was necessary. After extensive consultation with Professor Mustaque Ahamad [the Practicum course's instructor], it was decided that an objective evaluation of tool efficacy could be obtained by presenting it to a panel of outside observers and collecting their feedback. It was decided that the panel would be composed of my colleagues at the *South Carolina Department of Revenue*. They would be well-suited to serve as such due to their extensive security experience and lack of prior knowledge of *VulnTraq*. A demo presentation and survey was prepared and responses were collected. This report's *Section 7* describes the evaluation's particulars.

5.4 Prototype Limitations – Development

The prototype was rapidly developed within the constraints of a compressed summer semester. The process of backend preparation through the installment and configuration of the UVDesk application, its API bundle, and *ngrok* started on May 20th and ended in the beginning of June. The prototype's main functionality is located in its frontend, developed from May 30 through July 13. Because the program was rapidly developed over the course of approximately 1.5 months with the goal of a minimal viable prototype, the frontend's code may not be optimal in terms of readability and algorithmic efficiency. The time constraints also precluded unit testing of the codebase with frameworks like *Selenium* to identify unaddressed bugs.

6. Future Work

The prototype can be improved in multiple aspects with continued work in the future.

6.1 Future Work – Code Refactoring

The prototype's main capabilities were implemented in its Vue.JS frontend. Due to it being rapidly developed within a 1.5 month timeframe, the frontend's code may not be optimal in terms of readability and algorithmic efficiency. An intensive code review could be performed to identify and remediate any inefficiencies in program execution and formatting.

It should be noted that its algorithmic efficiency significantly depends on the capabilities provided by the backend – or the lack thereof. This report's section 5.1 relates an example of how insufficient options for creating a new ticket with all desired field values using the **POST /api/v1/ticket** REST API instruction resulted in an extensive workaround. The need for similar inefficient and time-consuming workarounds could be eliminated by pulling a branch from the open-source UVDesk project, developing custom REST API instructions, and repackaging the final product into a final *VulnTraq-Backend* release for use in enterprises' servers. [14]

6.2 Future Work – Additional Capabilities

The prototype could be improved to provide greater value for current and future users with the following capabilities:

- Supporting user account creation, logging in, and logging out functionality. Each account will also be tied to a valid and active email address for the user to receive notifications with
- Automated notification alerts in the user interface and via email to ticket owners for upcoming patching deadlines
- Presenting a separate website page containing reports instead of a pop-up modal when the frontend's *Open Reports* button is clicked. A larger size of the presented window will allow for more text and data to be presented
- Generating and presenting new metrics and graphs to meet the needs of a vulnerability program's executive, operational, and information security stakeholders and their needs. These include those drawn from the SANS Institute's *Key Metrics: Cloud & Enterprise*, listed in the report's *Appendix C*
- Automatically generating reports on a user-specified timetable [once per day/X days/week/month] and emailing them to end users through a designated mailing list
- Allowing users to export presented table information and reports as .CSV and Excel files
- Integrating it with open source vulnerability scanners (like OpenVAS) to automatically create new patching tickets on discovered and to verify that previously-closed tickets were properly closed

6.3 Future Work – Evaluation

The prototype can be evaluated further to identify points of weaknesses and drive future improvements. This ideally be done by collecting metrics for multiple enterprises' vulnerability programs in their current operational states, field testing *VulnTraq* in operational environments over an extended time period (e.g. 6 months to a year), and collecting user feedback and a second set of metrics.

If such field tests are not possible, its current and future iterations can be evaluated by presenting it to a larger information security professional audience and collecting feedback through a survey. The audience could be sourced through a combination of social media posts on LinkedIn and Twitter and information security conferences and meetups. Such would result in a greater range of feedback due to a larger variety of respondents' professional experience and skills.

6.4 Future Work – Deployment

The prototype's frontend and Virtual Machine backend were developed and tested within the local context of a single workstation. To prepare *VulnTraq* for operational deployment in an environment, the Vue.JS frontend needs to be deployed into a hosting service like Digital Ocean with the virtualized backend server deployed either into an on-premise or cloud-hosted server. This will allow multiple users in an enterprise to work with the web application simultaneously. It will also allow the much-needed testing of what happens when multiple website instances

simultaneously interact with the backend server. Identified issues stemming from simultaneous multi-user usage can then be promptly remediated.

6.5 Future Work – Fostering Collaboration With The Open Source Community

VulnTraq was envisioned as a tool to be grown and developed over the years by the open source community. To ensure this vision's achievement and grow the base of *VulnTraq*'s developers beyond one person, it is imperative to network with the open source and hobbyist programmer communities and attract people to the project. A lack of such a group of dedicated supporters will harm *VulnTraq*'s long-term potential.

6.6 Future Work – Security

The prototype was developed within a 1.5 month timeframe. With all focus directed at the rapid implementation of the product's capabilities and features, there has not been any effort to test the new codebases' security. Accordingly, this prototype and future versions need to be regularly inspected and pentested to discover vulnerabilities that need to be remediated.

My software security experience is limited to the knowledge of major vulnerability classes and types. I have never been on a pentesting engagement, much less that of a web application. I am thus not able to relate in great detail *how* such would be done. I do know that UVDesk's REST API would warrant extensive examination and that the frontend can be tested with Burp Suite.

7. Evaluation

Because field-testing the *VulnTraq* prototype in an enterprise was not possible during the compressed Summer 2022 semester, another evaluation method needed to be used. After extensive consultation with Professor Mustaque Ahamad [the Practicum course's instructor], it was decided that its functionality could be best done by presenting it to an outside panel of information security professionals.

Because I personally know a low number of information security professionals apart from my colleagues at the *South Carolina Department of Revenue* (my current employer), I decided on them as my panelists. I knew from my personal interactions of their extensive expertise and experience in a broad variety of operational settings and was confident that their responses would be useful and constructive. Accordingly, I created a thirty-minute YouTube video demo and collected their responses. The survey was given in the July 14-16 timeframe with four respondents in total. One is a SOC manager and the other three are security analysts.

The reactions to the presented prototype were generally positive. Respondents particularly appreciated its capacity for automatically generating reports and graphs and offered constructive suggestions for improvement. The reactions indicate that *VulnTraq* is a promising solution for the security industry.

This report's **Appendix B** lists the asked questions and raw responses.

Appendices

Appendix A: *VulnTraq*'s Frontend

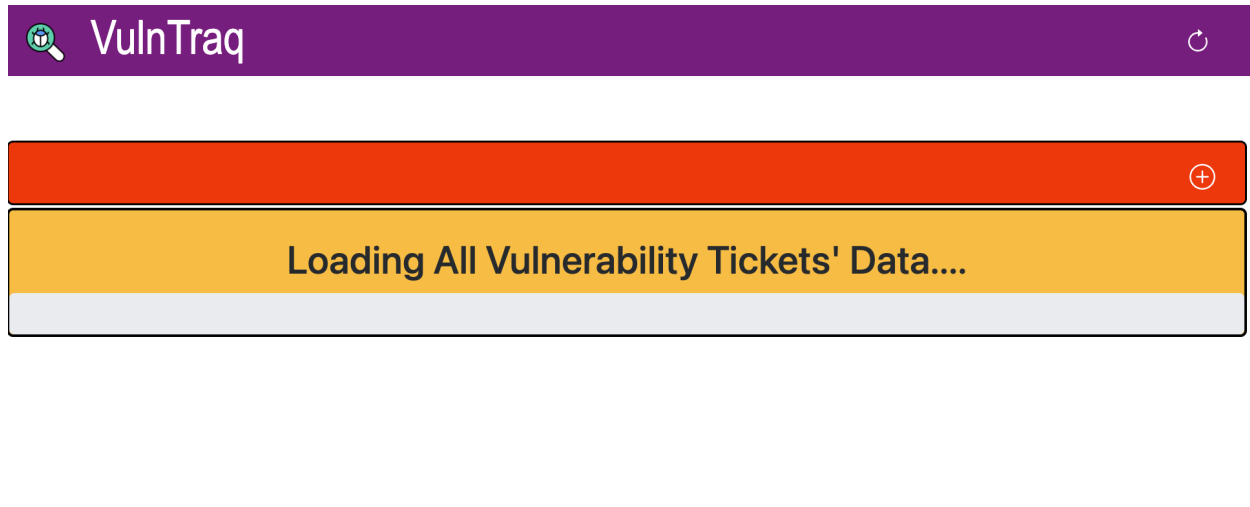
The frontend's code is available at the <https://github.com/secnate/VulnTraq> Github repository's *vulntraq-frontend* folder. It should be noted that its *main.js* file is home to the lines configuring

the information needed for successful communication with the UVDesk backend through its API: its URL and API key.

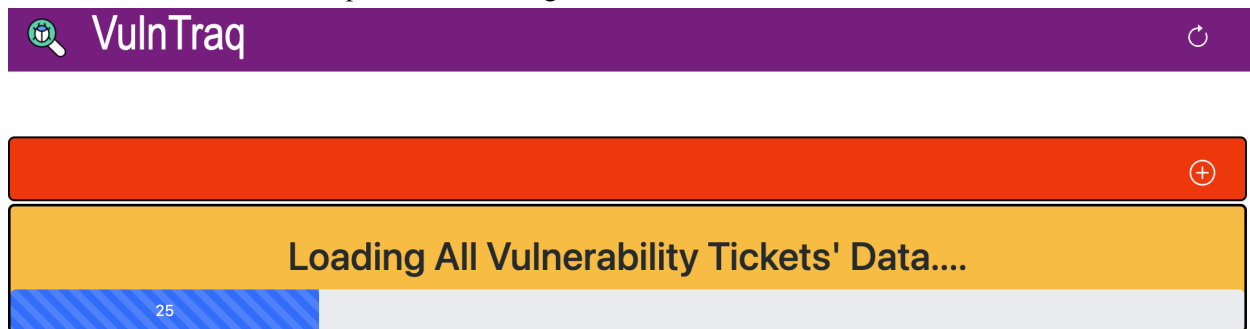
Demo of the application's frontend and its capabilities: https://youtu.be/pyRpl-hPn_4

Frontend Screenshots

Screenshot 1: The VulnTraq frontend was just opened in the browser, verified backend connectivity, and is starting to extract and load all tickets' data from the UVDesk backend server.



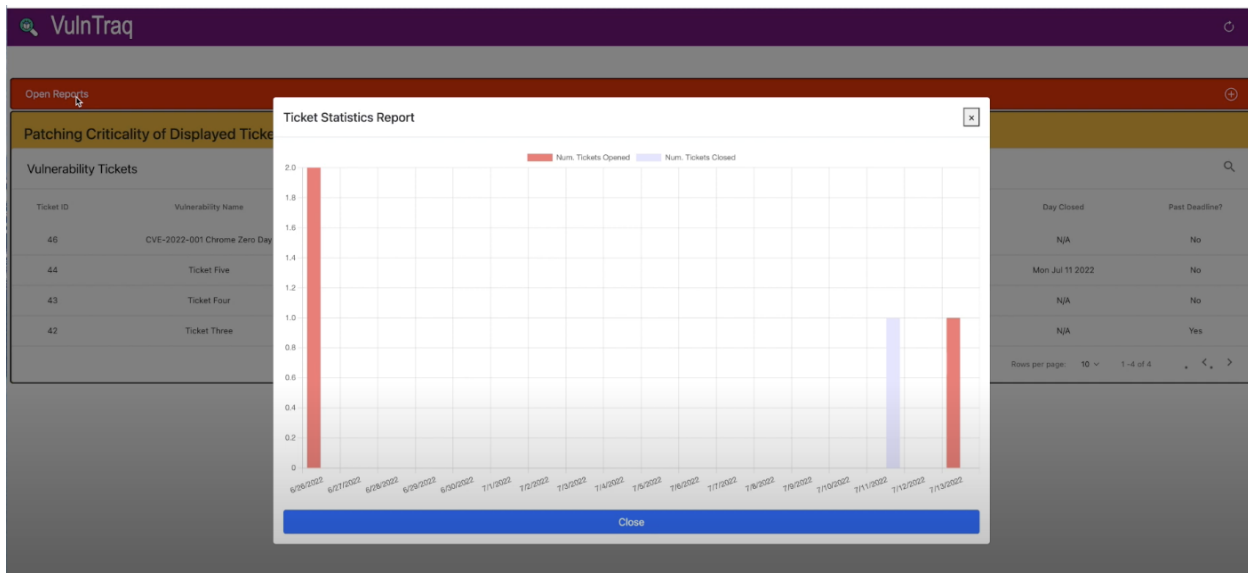
Screenshot 2: The VulnTraq frontend loading all tickets' information from the UVDesk backend server



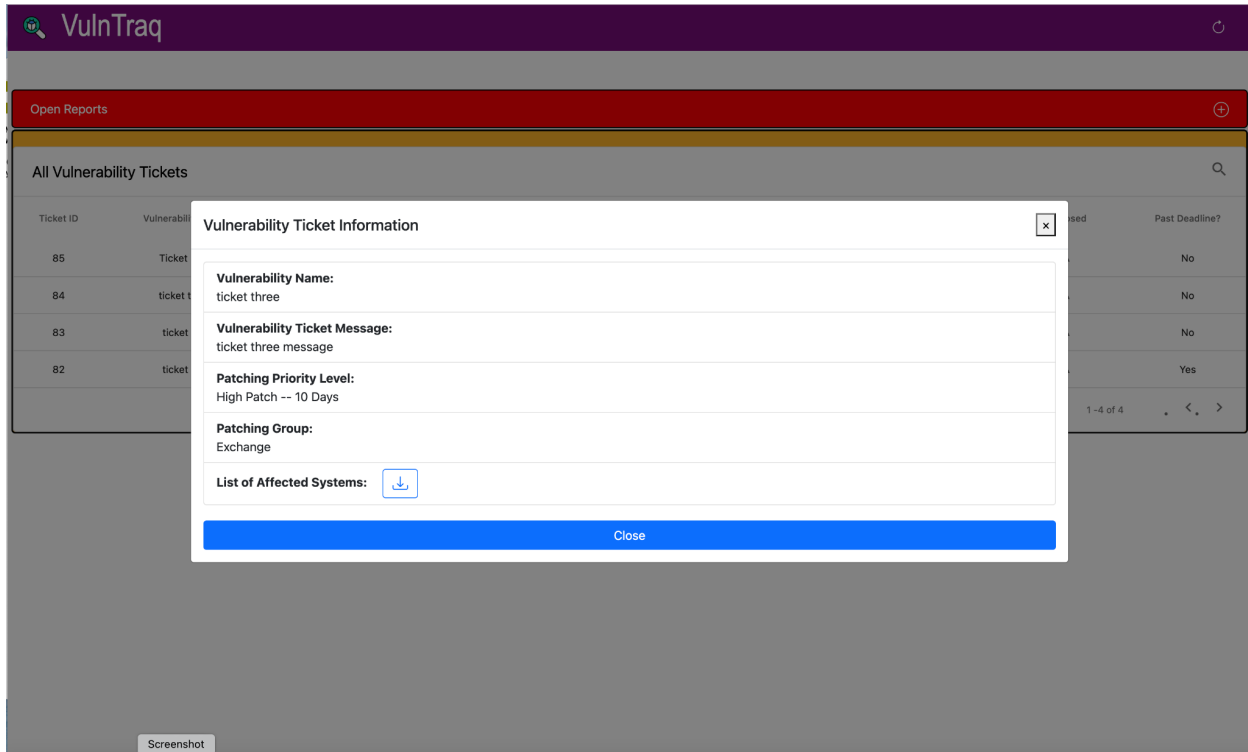
Screenshot 3: Demoing the VulnTraq frontend displaying all previously-opened tickets. The information is displayed after the VulnTraq application is loaded into the browser, verifies backend connectivity, and extracts and processes all involved data from the backend.

VulnTraq								
Open Reports								
All Vulnerability Tickets								
Ticket ID	Vulnerability Name	Status	Patching Group	Priority	Day Ticket Created	Due Date	Day Closed	Past Deadline?
85	Ticket Four	Open	Browsers	Medium Patch -- 30 Days	Tue Jul 05 2022	Thu Aug 04 2022	N/A	No
84	ticket three	Open	Exchange	High Patch -- 10 Days	Tue Jul 05 2022	Fri Jul 15 2022	N/A	No
83	ticket two	Open	Database	Medium Patch -- 30 Days	Tue Jul 05 2022	Thu Aug 04 2022	N/A	No
82	ticket one	Open	Database	Urgent Patch -- NOW	Tue Jul 05 2022	Wed Jul 06 2022	N/A	Yes
Rows per page: 10 1 - 4 of 4								

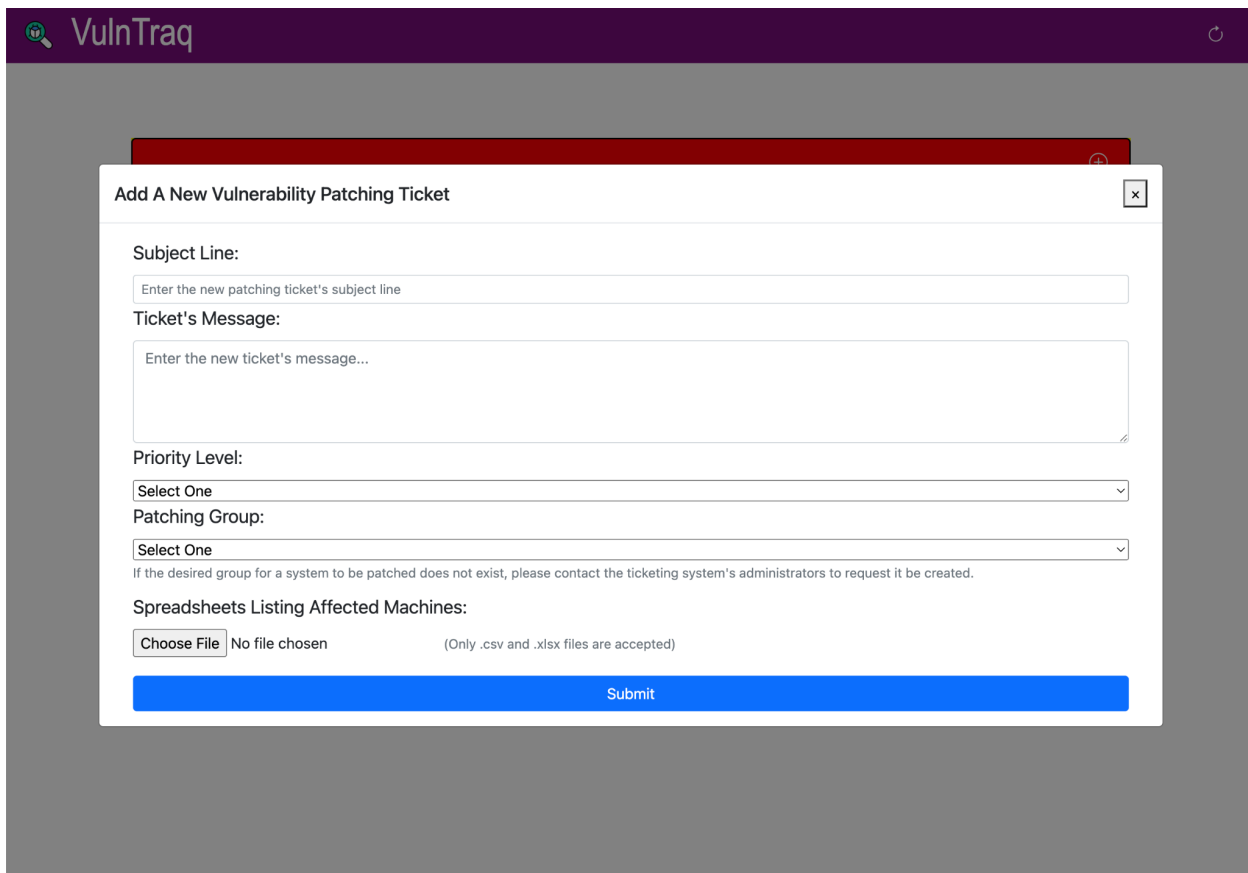
Screenshot 4: Displaying the generated report of the number of tickets opened and closed per day



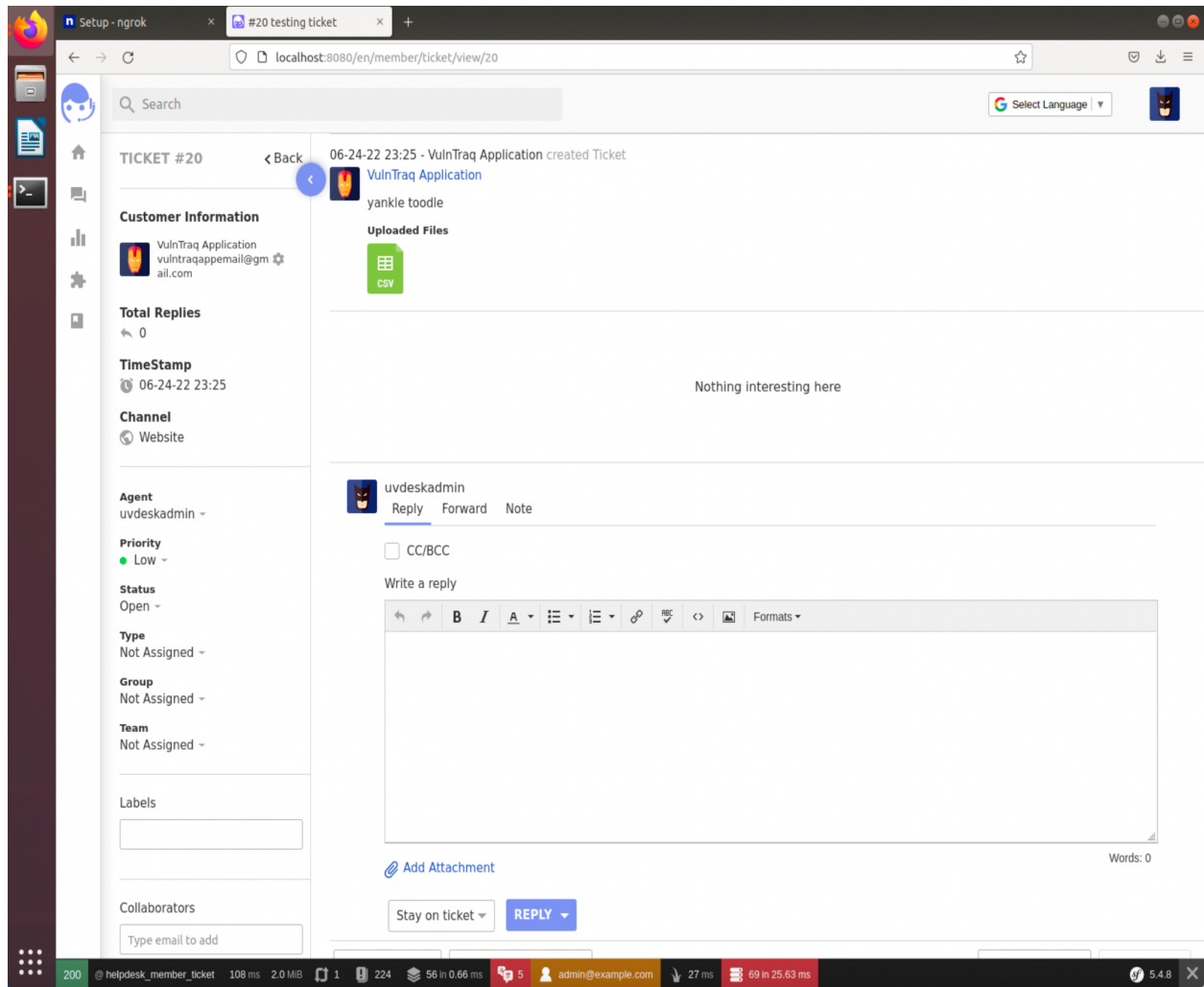
Screenshot 5: Displaying all the information for a selected ticket in a pop-up modal.



Screenshot 6: Demoing the vulnerability ticket-creating modal.



Screenshot 7: Displaying a sample ticket in the UVDesk backend server that was created by the vulnerability-ticket-creating modal's form being submitted. Notice the attached .csv file, which ticket creators need to include to list the systems affected.



Screenshot 8: Opening the *VulnTraQ* frontend with the UVDesk server Virtual Machine not up and running.



Backend Is Currently Not Accessible At
<https://7574-75-183-154-210.ngrok.io>

Clicking the top right-hand corner's refresh button without getting the UVDesk backend server up in the meantime leaves the presented page unchanged. If the server is brought online before the top right hand-corner's refresh button is clicked *and* the button is clicked afterwards, the application looks like *Screenshot 9* if there were no tickets created or like *Screenshot 3* if there were.

Screenshot 9: Opening the *VulnTraq* frontend with the UVDesk backend server up and running and no tickets having been created at all.



There Is No Vulnerability Ticket Information
Click The ⊕ Button To Address A New Vulnerability!

Appendix B: VulnTraq's Evaluation Feedback

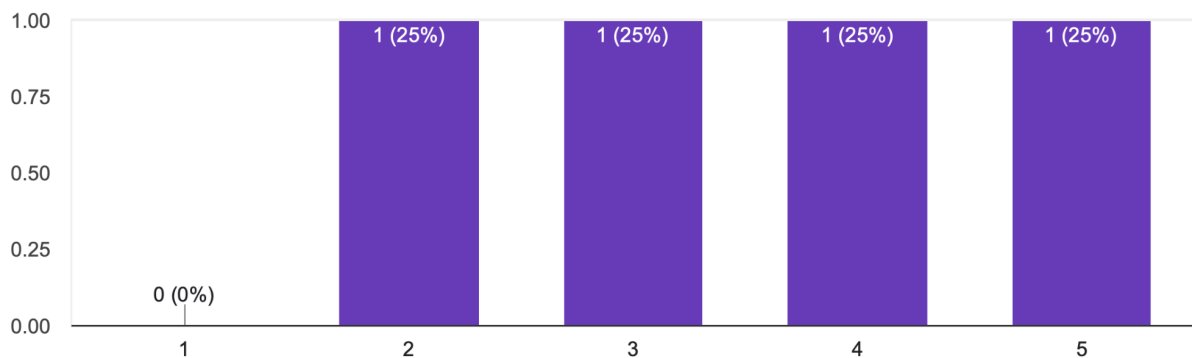
This Appendix details the questions asked to an outside evaluation panel after a thirty-minute video demo of *VulnTraq* and the collected survey responses. The survey was given in the July 14-16 timeframe with four respondents in total. One is a SOC manager and the other three are security analysts. The video demo presented to the panel can be found at https://youtu.be/pyRpl-hPn_4.

The following is a listing of the survey's questions and raw responses. Additional notes to raw responses are occasionally inserted to clarify certain terms and concepts that respondents used.

- **Survey Question #1:**

On a scale of 1 - 5 (Never to Always), how often do you work with vulnerability management on a daily basis?

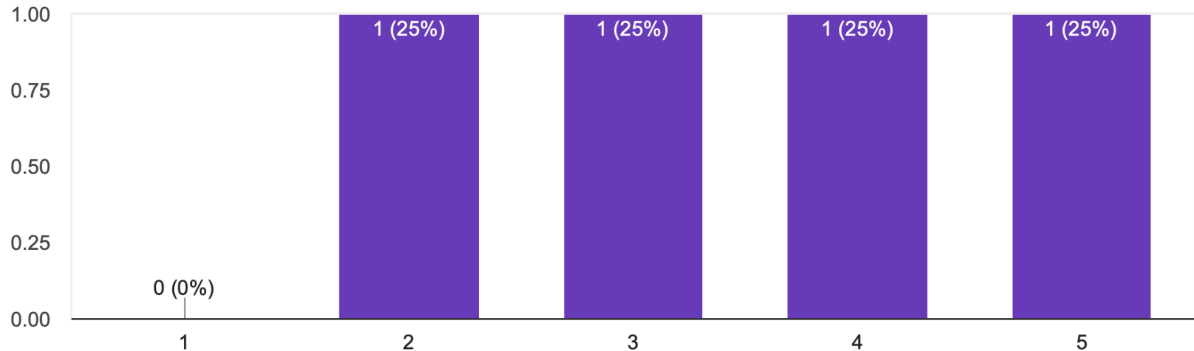
4 responses



- **Survey Question #2:**

On a scale of 1 - 5 (Never to Always), how often do you find yourself frustrated with the commonly-used Excel-based method of vulnerability management?

4 responses



- **Survey Question #3:**

What exactly frustrates you regarding the commonly-used Excel based method of vulnerability management? The answer's format is free response and you can respond however you want.

4 responses

too many files

This is a biased question. I'm not frustrated with excel but would lean toward a structured DB solution backend with a web front end with good reporting capabilities for use in a Vulnerability Tracking System.

Tracking vulnerabilities in Excel isn't a huge turnoff but as someone who used to be a Vulnerability Management Engineer, a GUI representation of the data is definitely preferred and with an accompanying ticketing system will definitely assist with the centralization of the vulnerabilities and their current statuses for both internal team tracking as well as management visibility.

If I use excel then I need to integrate it with other sources of data to make the latest scan report meaningful. Disparate data would benefit with the use of a single key that we could pivot with as well

Clarification Note: the "DB" acronym used in the second response means "database"

- **Survey Question #4:**

Have there been any tools you **have used** or **considered** in the past for performing vulnerability management and tracking ongoing vulnerability-related statuses apart from the typical method of using an Excel spreadsheet on a shared drive?

4 responses

Nessus

I have used Nessus and OpenVAS. The demo assumed that a ticket closure was accurate. Performing ongoing vulnerability testing as part of an automated solution would be ideal for independent validation of the threat management lifecycle.

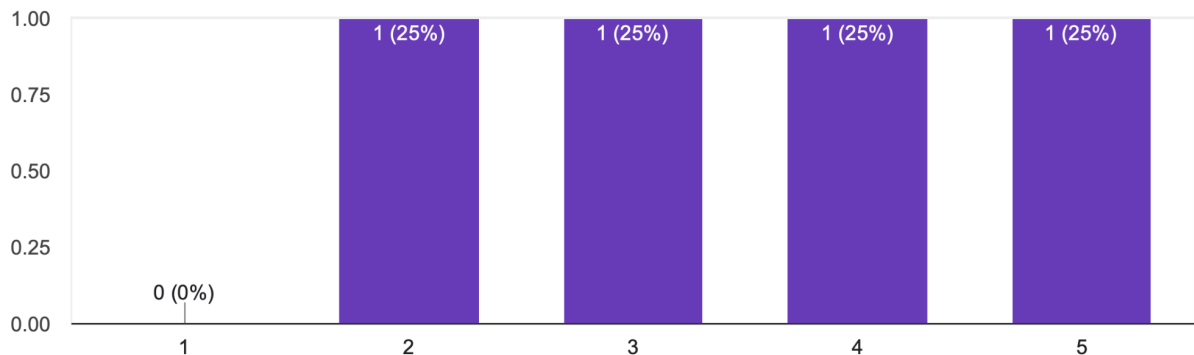
I was an administrator of a tool named Foundstone and rather than provide System Administrators with excel documents or deal with user management inside of Foundstone, we chose to create a separate web front end that tied into the back end database that tied into ldap for user management. Using this methodology, we were able to escape using an Excel spreadsheet.

We have home grown tools as well are reviewing an industry standard product

- **Survey Question #5:**

On a scale of 1 - 5, how often do you envision yourself using a tool like VulnTraq? Doesn't have to be VulnTraq specifically

4 responses



- **Survey Question #6:**

What VulnTraq capabilities would you consider the most important?

4 responses

The ability to prioritize

Reporting, and I would hope that it would have deeper data structure to support mitigations, exceptions, closer tie ins to

Accurate tracking, ldap integration, allow tickets to be assigned to specific users.

Automation to present the day in different ways so I could drill down and make observations

Clarification Note: *LDAP* is the *Lightweight Active Directory Protocol* used for authenticating users and granting them access to Active Directory-approved services in an environment. The mentioned desired *LDAP* integration aligns with the same respondent's response to *Survey Question #4* specifying a desire for it to be used for managing users

- **Survey Question #7:**

What VulnTraq capabilities would you consider the least important? Are they necessary?

4 responses

Nothing at the moment.

The front end, I would prefer to have it in a web application connected to a structured DB that would be homogenous for data sources to ingest relevant information into.

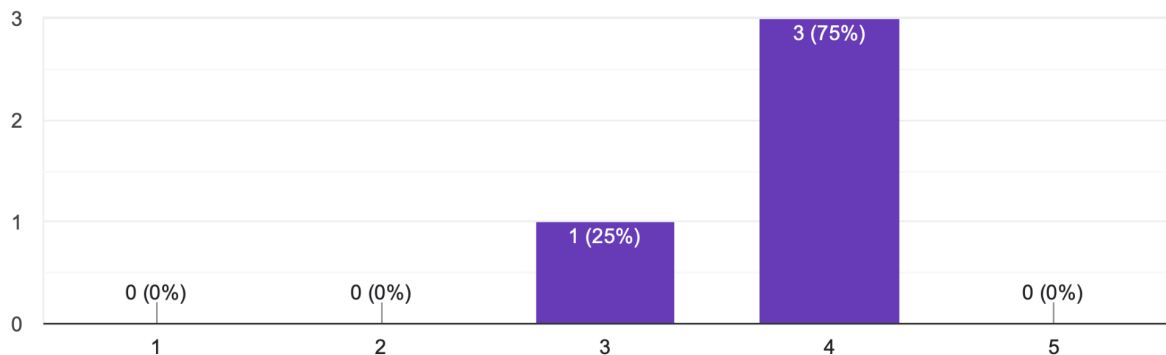
None at this time without further use and a POC.

None at this time

- **Survey Question #8:**

How easy do you find the application's user interface to use?

4 responses



- **Survey Question #9:**

What is your least favorite part of the VulnTraq application?

4 responses

Nothing at the moment. Could be more visually appealing?

Reporting, it needs more data points to drive it to Executive, Operational and Info Sec business needs. An Executive may want KPIs related Average Exposure Window, and Operational Staff may want to know about Vulnerability Churn Rate, while info Sec might want to see Exclusions or Intrusion attempts against specific vulnerabilities.

I think to find a feature that I didn't necessarily care for would (if application) come with use. The dashboard where you need to refresh each time may benefit from an option that allows for auto-refresh and I may have missed all of the options in the drop-down but if it allowed for searches such as 'tickets assigned to me', 'tickets assigned to my team', 'high/med/low/info severity tickets', etc., it may be beneficial.

I'd have to say the actual open source would be a hard sell for our organization. We shy away from open source.

- **Survey Question #10:**

What is your favorite part of the VulnTraq application?

4 responses

Can make a ticket pretty easy

Capex plus the required staff for development and implementation and maintenance.

I like both the simplicity and accuracy associated with VulnTraq. Just because every other tool is presenting with a bunch (sometimes unnecessary) features, it doesn't mean it's doing more...it just means you're paying for fluff development.

Automation

- **Survey Question #11:**

The presented VulnTraq application is a prototype. What would you want to see improved?
Any other capabilities you would want to add?

4 responses

Short cartoon animation 2min video to show how it works :)

Alignment with the SANS Key Metrics: Cloud and Enterprise and a function for mapping against the Vulnerability Management Maturity Model.

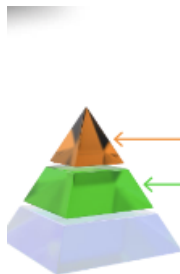
If it has the capability to accept API calls then perhaps automated calls from a VA system could automatically open tickets for tracking purposes without needing to manually open them. Perhaps the dashboard could have some widgets added that would allow for a graphical representation of the data above the fold.

Would need to run some pen tests on the product to make sure secure development practices are being used.

Appendix C: SANS Key Metrics to Be Incorporated Into Future Versions

The SANS Institute's *Key Metrics: Cloud & Enterprise* are a list of metrics that organizations can use to create their security programs and improve their maturity. Reports using the following vulnerability-specific metrics can be generated in future *VulnTraQ* versions to meet the needs of management, business, and operational stakeholders [15]:

- **Vulnerability Remediations Past Due Date:**



R Vulnerability Remediations Past Due Date

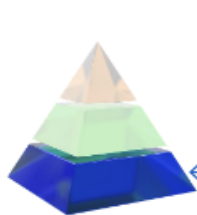
Advanced programs

DESCRIPTION
The remediations that are not meeting corporate policy requirements for remediation efforts that have not been granted an exception

HOW TO CALCULATE
(current date – first discovered date) > policy requirement (or if available, leverage due date field)

WHAT IT HELPS SHOW/IDENTIFY
Any remediation effort not meeting corporate requirements helps to show if there is a problem system or component, or potentially unrealistic remediation timeframes.

- **Patch Velocity:**



P Patch Velocity

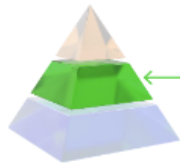
Advanced programs

DESCRIPTION
Patch Velocity counts patches applied per day.

HOW TO CALCULATE
ABSOLUTE VALUE (Patches applied on each date when the host was patched)

WHAT IT HELPS SHOW/IDENTIFY
This helps inform the organization how many patches were applied on each date when the host was patched. It can serve as a way to measure how frequently patching is happening in the environment.

- **Mean Time to Detect:**



D Mean Time to Detect

Advanced programs

DESCRIPTION

Mean Time to Detect is the average time it takes the organization to discover a vulnerability from when it is first published, or the asset is added to the network.

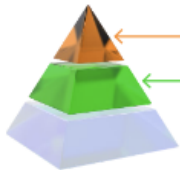
HOW TO CALCULATE

AVERAGE (Vulnerability Publish date – Vulnerability Discovered date)

WHAT IT HELPS SHOW/IDENTIFY

This metric gives you information on the exposure that the organization has due to vulnerabilities that exist but have not yet been discovered.

- Average Exposure Window:



I Average Exposure Window

Advanced programs

DESCRIPTION

The Average Exposure Window is meant to show how long the vulnerabilities are known about prior to them being remediated.

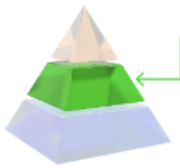
HOW TO CALCULATE

AVERAGE (Vulnerability Closed date – Vulnerability Published date)

WHAT IT HELPS SHOW/IDENTIFY

It helps track performance against the policy standards for various vulnerabilities. The goal is to have this as close to Mean Time to Resolve as possible.

- Vulnerability Reopen Rate by XXX:



D Vulnerability Reopen Rate by XXX

Advanced programs

DESCRIPTION

Number of vulnerabilities within the environment that are being re-opened for any reason. (XXX can be specific systems, applicaiton, business owners, administrators)

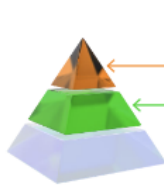
HOW TO CALCULATE

Number of vulnerabilities that were previously closed

WHAT IT HELPS SHOW/IDENTIFY

Identifies vulnerabilities that were felt to be addressed that no longer are, that normally point to a remediation system problem or a unique system

- Vulnerability Churn Rate:



P Vulnerability Churn Rate

Advanced programs

DESCRIPTION

Vulnerability Churn Rate is the rate that vulnerabilities are being closed as well as new vulnerabilities being opened

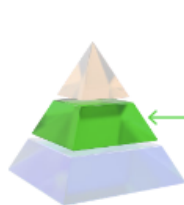
HOW TO CALCULATE

ABSOLUTE VALUE (New Vulnerabilities – Closed Vulnerabilities [over specific period of time e.g., monthly])

WHAT IT HELPS SHOW/IDENTIFY

It shows if the vulnerability management program is making headway or is losing the battle.

- Mean Time to Resolve:



R Mean Time to Resolve

Advanced programs

DESCRIPTION

Mean Time to Resolve is the average time it takes the organization from discovering a vulnerability until the vulnerability is remediated.

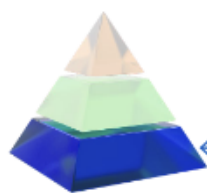
HOW TO CALCULATE

AVERAGE (Vulnerability Closed date – First Discovered date)

WHAT IT HELPS SHOW/IDENTIFY

This informs the organization how long it is taking from the time a vulnerability is discovered until it is remediated. It can provide insights when new vulnerabilities arise and/or how long until these are validated findings using normal processes.

- Vulnerability Scanner Coverage:



I Vulnerability Scanner Coverage

Early stage programs

DESCRIPTION

Vulnerability Scanner Coverage is the percentage of the system within your organization that is regularly scanned for vulnerabilities.

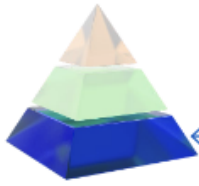
HOW TO CALCULATE

Assets being scanned for Vulnerabilities/Total Assets

WHAT IT HELPS SHOW/IDENTIFY

Knowing if systems are not regularly scanned is crucial to understanding the risk to the business and trend reports will not be as meaningful until coverage is stable.

- Patch Age:



P Patch Age

Advanced programs

DESCRIPTION

Patch Age of a system is the number of days since the last patch was applied.

HOW TO CALCULATE

ABSOLUTE VALUE (The number of days which have elapsed since the last time a patch was installed on the system)

WHAT IT HELPS SHOW/IDENTIFY

This helps inform the organization of whether patching has happened recently. Stakeholders can understand the number of days which have elapsed since the last time a patch was installed on the system. A low Patch Age does not necessarily mean that the system is fully patched, but it does indicate that some patching activity has taken place recently.

Sources

1. Truta, Filip. "60% Of Breaches in 2019 Involved Unpatched Vulnerabilities." Security Boulevard, Security Boulevard, 31 Oct. 2019, <https://securityboulevard.com/2019/10/60-of-breaches-in-2019-involved-unpatched-vulnerabilities/>.
2. Tunggal, Abi Tyas. "What Is the Cost of a Data Breach in 2022?: Upguard." UpGuard, UpGuard, 12 May 2022, <https://www.upguard.com/blog/cost-of-data-breach>.
3. "Majority of Small Companies Lose Business after Undergoing Cyber-Attacks." The Council of Insurance Agents & Brokers, <https://www.ciab.com/resources/majority-small-companies-lose-business-undergoing-cyber-attacks/#:~:text=Majority%20of%20Small%20Companies%20Lose%20Business%20after%20Undergoing%20Cyber%2DAttacks,-SHARE&text=Research%20from%20the%20National%20Cyber,to%20recuperate%20in%20the%20aftermath>.
4. Naumova, E. (2021, December 20). 5 trends businesses to consider when planning 2022 cybersecurity budgets. Retrieved July 13, 2022, from <https://www.ciodive.com/spons/5-trends-businesses-to-consider-when-planning-2022-cybersecurity-budgets/611073/>
5. Bitglass Cloud Security Report 2015, Bitglass, <https://pages.bitglass.com/Cloud-Security-Report-2015-PDF.html>.
6. "Alienvault (AT&T Security) USM Anywhere Vulnerability Management & Siem." Alienvault (AT&T Security) USM Anywhere Vulnerability Management & SIEM - Digital Marketplace, <https://www.digitalmarketplace.service.gov.uk/g-cloud/services/488014245528024>.
[service.gov.uk/g-cloud/services/488014245528024](https://www.digitalmarketplace.service.gov.uk/g-cloud/services/488014245528024).
7. *Features: Manageengine vulnerability manager plus*. Features | ManageEngine Vulnerability Manager Plus. (n.d.). Retrieved July 16, 2022, from <https://www.manageengine.com/vulnerability-management/features.html>
8. *Edition comparison matrix: Manageengine vulnerability manager plus*. Edition Comparison Matrix | ManageEngine Vulnerability Manager Plus. (n.d.). Retrieved July 16, 2022, from <https://www.manageengine.com/vulnerability-management/edition-comparison.html>
9. Kibet, J. (2021, October 20). *Setup UVdesk ticketing system on ubuntu 20.04*. ComputingForGeeks. Retrieved July 18, 2022, from <https://computingforgeeks.com/setup-uvdesk-ticketing-system-on-ubuntu/>
10. Uvdesk. (n.d.). *Uvdesk/API-bundle: API integration for UVDESK Community Helpdesk System*. GitHub. Retrieved July 18, 2022, from <https://github.com/uvdesk/api-bundle>
11. Atta. (2020, February 21). *How to use axios to send HTTP requests in Vanilla JavaScript*. Atta. Retrieved July 18, 2022, from <https://attacomisian.com/blog/axios-javascript>
12. *Ticket related apis · uvdesk/API-bundle wiki*. GitHub. (n.d.). Retrieved July 19, 2022, from <https://github.com/uvdesk/api-bundle/wiki/Ticket-Related-APIs>
13. Webkul Software Pvt Ltd. (2018, June 15). *(English) rest apis documentation: UVdesk Developer Portal*. UVdesk Helpdesk. Retrieved July 19, 2022, from

<https://www.uvdesk.com/en/api-doc/>

14. YouTube. (2020, March 4). *How to create a custom API in UVdesk Open Source Helpdesk*. YouTube. Retrieved July 19, 2022, from <https://www.youtube.com/watch?v=m-zrtOuGwpU>
15. *Sans cybersecurity leadership*. Key Metrics and Vulnerability Management Maturity Model Poster | SANS Institute. (2022, May 16). Retrieved July 23, 2022, from <https://www.sans.org/posters/key-metrics-cloud-enterprise-vmmm/>