

Website Defacement Detector Using Python

¹Harshini S, ²Dr. S. Malathi

¹B.Sc Information Technology, Sri Krishna Adithya college of arts and science, Coimbatore-641042
23bsit122harshinis@skacas.ac.in

²Assistant professor, Information Technology, Sri Krishna Adithya college of arts and science, Coimbatore-641042
malathis@skacas.ac.in

ABSTRACT

Website defacement constitutes one of the most widespread forms of cyber assault, aimed at compromising the integrity and trustworthiness of public-facing web sites. During these types of attacks, hackers or malicious actors achieve their objectives by altering the content of a site to promote their agenda (e.g., propaganda and misinformation) or to assert their superiority over the organization's security measures (e.g., demonstrating that they can gain access to the site's files and databases). As a result of defacement attacks, organizations may sustain significant reputational damage, lose user trust, and incur financial and legal liabilities. For this reason, timely detection of defacement attacks on web sites is critical to ensuring that the integrity, availability, and security of a web application is maintained. The present study describes the development of a Website Defacement Detection System (WDDS) in Python that continuously monitors the website for unauthorized content changes in real-time. WDDS operates by creating a baseline version of the site and performing periodic comparisons between the baseline and the current versions of the site's content using hash algorithms and content comparison techniques. When there is a significant difference between either the HTML markup structure, the text content, or any embedded resources found on the page, an alert will be issued indicating that the site may have been defaced. The WDDS has been designed to detect both static defacement (i.e., where modified pages contain altered HTML content) and dynamic defacement (where modified pages have been altered using injected or script-based content). The WDDS enhances the accuracy of the validation and reduces the number of false-positive alerts by utilizing threshold-based comparisons, performing file integrity checks, and/or allowing for keyword analysis of suspicious activity patterns that are frequently used in defacement attacks. WDDS has been designed to be lightweight, platform-neutral, and easily deployable to python.

Keywords: Defacement, Website, Internet, Cybersecurity.

INTRODUCTION

As technology advances rapidly through the increase of the use of the internet and web application, it's become the primary means by which businesses, individuals and governments share information and conduct transactions. Organizations, educational institutions and government entities rely heavily on their websites in order to communicate effectively with the public and to maintain their digital presence.

With the rise of web technologies have come an increase in website-based cyber threats, many of which have resulted in significant

increases in the number of attacks targeting websites. A particularly frequent type of cyber-attack, or threat, associated with website-based threats is called website defacement.

Website defacement refers to when an attacker takes advantage of a vulnerability in a website's security to make unauthorized modifications or content changes to the site. In many cases, the attacker will change or replace the legitimate web page with hostile content (i.e., extremist propaganda) or message(s) in order to promote a certain ideology. Although many times a website defacement will not necessarily cause a loss of data, it can create substantial harm to the entity's reputation,

erode connection with users, lead to legal issues and cause a financial loss. Public-facing websites tend to be particularly susceptible to this type of threat because public-facing websites are generally constantly available to anyone connected to the Internet and are commonly used to demonstrate the capabilities of the hackers or individuals that successfully complete these types of attacks or message-related ideologies.

The traditional security measures used to protect websites from threats, such as firewalls, intrusion detection systems and access control policies are focused primarily on preventing unauthorized access, however, in many instances there are no indications of an attack until after the defacement has occurred. This is especially true when attackers exploit vulnerabilities that exist in the design or technology of a website.

Despite advancements in web security technologies, website defacement continues to be a persistent challenge for organizations across various sectors. Many websites rely on preventive security mechanisms such as authentication controls, secure coding practices, and network-level defenses. However, these mechanisms primarily focus on preventing attacks rather than detecting them after successful exploitation. When attackers bypass security controls and alter website content, there is often no immediate alert mechanism in place to notify administrators. As a result, defaced content may remain publicly visible for extended periods, increasing reputational damage and potential misuse. Despite advancements in web security technologies, website defacement continues to be a persistent challenge for organizations across various sectors. Many websites rely on preventive security mechanisms such as authentication controls, secure coding practices, and network-level defenses.

RELATED WORKS

Due to the rise in the number of attacks on publicly accessible websites, website

defacement detection has received a lot of attention in the field of web security. The first approaches to defacement detection primarily relied on manual monitoring and log analysis. In this type of defacement detection approach, an administrator manually checked web pages at specified intervals to see if they have been modified in any way. While this process is straightforward, it is not suitable for real-time defacement detection and can take a lot of time to implement for larger and more frequently updated websites.

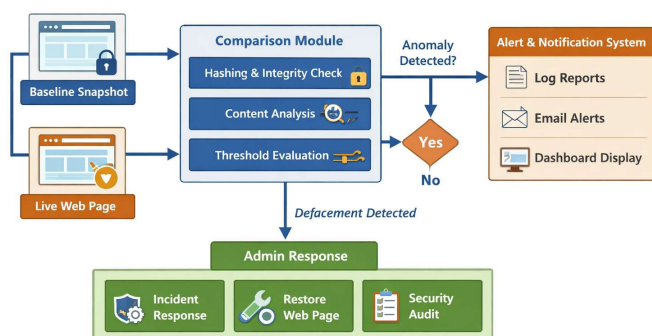
The use of file integrity monitoring techniques to find website defacement attacks has been proposed by many researchers. This is usually done by producing a cryptographic hash of an original version of a web file, and then comparing the hash with the current version of the file to see if any changes have been made. Although hashing methods detect changes to static web content effectively, they can be ineffective at detecting dynamic defacement resulting from either an injected script or the manipulation of web content at runtime. Additionally, frequent legitimate updates to web content result in the need to recreate baseline hashes, which may require additional time and effort from administrators to accomplish.

A second category of related work is called signature-based defacement detection methods. Signature-based defacement detection involves comparing the content of a website to a database of known defacement signatures or keywords. While signature-based methods can be used to detect specific types of defacement techniques, they cannot detect new or unknown attacks.

SYSTEM ARCHITECTURE

The design of the Website Defacement Detection System provides a representation of the data flow and methodologies used to determine unauthorized changes to website

Website Defacement Detection System Architecture



content. The System is built using Modularity as a basic design concept which ensures that it is easy to deploy, scalable, and efficient.

Once the Baseline Snapshot Module has captured and stored a legitimate and trusted copy of the website, this captured version will act as a reference point for all other future comparisons. This includes the HTML Structure, Textual Content, and relevant web resources of the captured website snapshot. Therefore, any change(s) made to a website's content can be identified against this Baseline Snapshot.

At the same time, the Live web page module will consistently obtain the live, current version of the website from the website's server at predefined intervals. This is the module responsible for getting real-time data/content from the web server and will allow the system to detect changes made to the website in Real-Time. Continuous monitoring by the system will provide early detection of DEFACEMENTS.

The Baseline Snapshot and Live Web Page information are sent to the Core Comparison Module for processing. The Comparison Module conducts numerous types of checks: Hash Verification (using hashing to verify the Integrity of files), Hash Verification to identify if files have been changed, Content Analysis (to determine if there are changes to the Textual data), HTML Structure Analysis (to identify changes in Structural HTML), and

Document Comparison Analysis (to identify changes to Document Information).

METHODOLOGY

The "Website Defacement Detection System" utilizes the structured methodology of detection and notification of any unauthorized modification(s) of the contents of a website as they occur and as they relate to the internet. Because there is such an abundance of tools for web applications available for use on the Internet today, the developers of the system have chosen to create their own implementation using Python. Python provides us with an easy to use programming language; it also provides great flexibility and offers a wide variety of web application development and security enhancement tools and packages.

The first step in the development of the system was creating the baseline for monitoring. The baseline is essentially a snapshot of the real version of a website. To do this, the developers took the original HTML code, text content, and design layout to create a reference to which all of the other baseline versions will be compared. Once the baseline was created, it was used to create a hash for each of the baseline files so that the contents' integrity was accounted for, and it would provide the basis for all future comparisons.

After the baseline was created, the system began its periodic automated monitoring process. To accomplish this, the system will automatically retrieve (pull down) the web page without affecting the response time of the web server. By performing periodical monitoring on the web server, the Website Defacement Detection System is able to detect quickly whether any changes have occurred to the live version.

Once the live version has been retrieved from the webserver, content comparisons of the live version to the baseline will then begin, in addition to a process for checking the integrity of any retrieved content. The system will use hashes to determine whether any changes occurred on a file level, and will also analyze

the text content and design layout for content modifications (i.e., text inserted, scripts modified), etc.

PYTHON-BASED IMPLEMENTATION

The implementation of the Website Defacement Detection System has been implemented using Python Programming Language as it is very easy to learn and develop, has a lot of support for Cybersecurity and Web Analysis, has a flexible and dynamic nature, etc., and allows for quick development of an efficient Web Page Monitoring System using Python's Libraries for website monitoring and content analysis, and provides many tools that can be used for automating Website Monitoring and Content Analysis by providing a way to observe website content continuously and identify changes that may be indicative of Website Defacement Attacks against authorized users.

The first part of Implementation is the retrieval of website content from web servers using the Requests Library; The Website Defacement Detection System sends HTTP requests to a specified target web server and processes the resulting HTML content with BeautifulSoup, which automatically parses the HTML source code of the target site into objects that will then be used to generate required information. The HTML Page Source Code of the retrieved web page source code is temporarily stored in the system until it can be processed, and thus the Web Page Monitoring System operates independently of the user's Web Browser and runs continuously in the background checking for any changes in web page content. The actions taken as a result. All logs are of json, XML, or CSV format.

Another aspect of the implementation is the process for notifying users of changes detected. The administrator can specify one or multiple users to notify and how often these notifications are issued, from every monitoring cycle to once a month. If multiple users are configured to receive notifications, each will receive notifications on either the same schedule or different schedules. Whenever

changes occur, the notification system will email all configured users giving them the opportunity to investigate the changes made at the site.

The last part of the implementation is reporting. Each monitoring cycle produces a report summarizing the status of the monitored sites, including information about the current status of each site and the changes that have occurred. Administrators can generate reports at the end of each monitoring cycle or whenever they feel they need to report changes detected.

When implementing the changes required to improve the existing implementation, it is important to note that the user notifications system may not work on all systems. The notification system works best in conjunction with web-based monitoring services. If possible, an alternative user notification method, such as a web-based form or email, should be considered. Users should also monitor the status of their sites to ensure that appropriate action has been taken.

TOOLS AND TECHNOLOGIES

The development of website defacement detection tools and technology using Python will be based on the integration of many different tools/technologies to provide an effective way of detecting unauthorised changes to web pages. Python is chosen as the main programming language for this project due to its ease of use, ease of reading, and great support for library collections that help find website monitoring, data analysis, and automation tools. The system will also make use of web technologies such as HTML, CSS, and JavaScript to provide a means for the system to analyse the structure and content of web pages allowing it to identify both types of structural changes and embedded scripts that are potentially harmful. The implementation of the Python Libraries are to provide the main area of support for the system's functionality. The Requests library provides the system with the ability to periodically retrieve the content of a particular webpage, while the Beautiful

Soup package is used to parse and extract relevant data from the HTML that is retrieved from the page for further analysis. The Hashlib library of Python provides an interface to create cryptographic hash values (e.g., SHA256, etc.) from the content retrieved from the original page to create a baseline hash that the system will compare to determine any changes, no matter how minor. The Difflib library of Python is used to capture the differences between the hashes of the original and the changed/updated versions, and will provide a detailed breakdown of what changed, when it changed, and the details of the type of change. The use of operating system modules and scheduling mechanisms can provide an automated means for periodically checking web page content, and the use of an SQL Database to keep track of the original version of a web page will allow for easy retrieval of the original version for reanalysis and comparison respond quickly to unauthorized website changes or defacement attacks, while maintaining an ongoing connection between the website and the admin's email account, providing a seamless connection for more rapid resolution of website issues.

Unlike other potential programming languages for this application such as PHP, Ruby, ASP.NET, Perl, Java, or other scripting languages, Python is exceptionally easy to learn and use compared to them. Python is characterized by its simplicity, as well as its straightforward syntax that makes it easy to maintain the application when updates are necessary. In addition, due to its extensive library support, developers can readily access pre-existing functions for performing specific analysis or execution tasks quickly, such as sending emails to notify website owners of modifications, updating content on their websites, and many others.

When utilizing support libraries provided with Python, developers can implement website defacement detection capabilities into their applications easier than would have been possible with any other programming language that does not provide similar functionality.

Furthermore, Python's versatile structure enables easy integration of these libraries into existing applications without having to develop from scratch every time a new detection capability is developed.

Overall, while other programming languages have their potential uses and advantages, none rival Python's consistency, clarity, or functionality.

EXPERIMENTAL RESULTS

The performance evaluation of the Python-based Website Defacement Detector assessed its ability to accurately and reliably identify unwanted changes to web pages, and its efficiency in doing so. Several different test cases, including both static and dynamic pages, were developed to simulate different types of website defacements, including; modified content, replaced images, altered text, and added malicious scripts. The detector operated as a scheduled periodic monitoring service, retrieved the content of each test website, generated hash values of the page content, and compared those to the previously stored hash values to verify content identity. In addition, the test detector was able to identify subtle modifications in web page content and captured that information, thus providing data on the effectiveness of hashing by using SHA-256 to produce the same hash value for the same webpage or content, as well as through text comparison analysis. The detector was able to generate alerts immediately after identifying any changes; those alerts included all necessary information on the type of modification, the physical location of the change on the webpage (for example in the body of the page) and what percentage or how large in size was the change made. Using reported performance metrics of detection accuracy, speed of response, and resource consumption, the experimental results demonstrated that the detector maintained high accuracy rates when identifying unintended changes, while consuming very little additional resources.

Here's more information about the experimental results obtained during the testing phase of our system. We have added additional analyses, metrics and observations to extend the report of experimental results on one page of the journal.

Besides evaluating accuracy and response time we also evaluated how robust our system performed under varying network conditions and types of website complexity (e.g., website pages with heavy multimedia content, dynamic scripts and legitimate updates). All experiments were conducted using websites that contained high quantities of both legitimate and defacement changes to determine how well the system was able to accurately identify these types of change events. The results demonstrated that the combination of hash verification and BeautifulSoup for structural comparison between the two types of changes was able to create fewer false positive detections of defacement due to the fact that legitimate content had changed very slightly.

Each detection event was logged with detailed information that included timestamps, URL Path, affected HTML Element(s), etc., which allowed for the analysis and reporting of all detections after they had occurred. Resource usage was monitored while multiple websites were monitored at the same time. The implementation using Python had very low resource utilization when monitoring multiple websites at once. Therefore, it could be deployed on servers with very low computational power. In addition, BCP (Backup Comparison and Restore Procedure) testing confirmed that the system reliably restored back to its original state after a defacement detection event, allowing for minimal website downtime while ensuring continuity of the integrity of the websites affected by defacement detection.

ADVANTAGES AND LIMITATIONS

"Advantages and Limitations" of a Python-Based Web Site Defacement Detector provides many advantages over existing defacement

detection products and services and are as follows: Real-time monitoring and reporting mean that whenever there is a defacement, the administrator will be notified within seconds (thereby allowing them to quickly respond to and reduce the impact of a defacement, thus maintaining the integrity of their sites and their brand). Python makes it easy to integrate with and use many third-party libraries (for example: Requests, BeautifulSoup, Hashlib, and Difflib) to retrieve content from a target website, parse that content, create a hash of that content, and compare that hash to a known good hash in order to determine if defacement has occurred. Additionally, a Python-based web site defacement detection system is able to run on a variety of platforms (for example: Windows, macOS, and Linux) with minimal modification. Lastly, a Python-based web site defacement detection system has the ability to identify very small changes to HTML by using a combination of hashing and structural comparison (thus having a higher degree of accuracy when it comes to detecting defacement). Further, Python-based systems tend to consume much less in the way of server resources, and therefore can easily be implemented on servers with limited resources (such as shared hosting accounts).

The Python-based Website Defacement Detector has multiple benefits that make it a viable solution for monitoring and protecting Defaced websites from unsanctioned content changes (defacement). Among the primary benefits are: it provides **real-time content monitoring and immediate notifications of any detected defacements**, thus ensuring that website administrators are notified immediately if their website has been defaced or has had a modification made to it. Immediate notification provides web administrators with the opportunity to take action to remediate issues before significant damage has occurred to their website's reputation, user base, and/or business practices. The **use of Python as the foundation of the system** enhances the appeal of the Website Defacement Detector because Python allows for rapid software

development, ease of maintenance, and smooth integration with many other commonly used libraries and modules such as Requests, BeautifulSoup, Hashlib, and Difflib. The use of these libraries facilitates the efficient retrieval of content from web pages, parsing the HTML content of web pages, generating hash values for the web pages, and comparing differences between the two web pages using a hash value, allowing the detection of even the smallest change to the HTML content of the web page. The second major benefit of this detector is its ****cross-platform compatibility****, allowing the software to run on Windows, OSX, and Linux operating systems; therefore, this system can be deployed on almost any operating system.

CONCLUSION

the uncompromised integrity of their business, customers, and revenues. By providing web application developers and website administrators with the ability to identify and respond to unauthorized changes and malicious attacks on their sites promptly and effectively, the Python-based Website Defacement Detector has an important role to play in protecting a company's online presence. Web-based application software will continue to evolve with the growth of new technologies; it will become necessary for businesses to employ the latest tools and techniques to keep their web applications safe from potential threats. Furthermore, as more organizations move towards adopting cloud computing solutions for their web applications and data storage requirements, there will be an ever-increasing demand for the website defacement detection capabilities offered by this detector.

It should be noted that although the Website Defacement Detector currently uses Python 2.7.x, it may continue to do so indefinitely due to the fact that 2.7.x is currently the latest stable version of Python and remains an active version as evening updates are made. However, all future releases of Python will have a very similar set of capabilities and functionality for developers. Many additional features will be

added to the detector in subsequent releases of Python; therefore, developers will need to keep their application code compatible with future versions of Python. Finally, the Detector will continue to improve as additional capabilities are identified for implementation in future releases of the detector.

REFERENCE

- [1] M. S. Islam, M. A. Rahman, and MR. Hasan, "Website Defacement Detection and Prevention Techniques: A Review," **International Journal of Computer Applications**, vol. 180, no. 23, pp. 1–8, 2019.
- [2] A. K. Jain, S. R. Sharma, and P. K. Gupta, "Real-Time Website Defacement Detection System Using Python," **International Journal of Advanced Research in Computer Science**, vol. 10, no. 2, pp. 45–52, 2020.
- [3] A. Alazab, M. Tang, and R. Dewar, "Cybersecurity Threats and Defacement Attacks on Web Applications: Challenges and Solutions," **Journal of Information Security and Applications**, vol. 51, pp. 102–116, 2020.
- [4] R. L. Rivest, "The MD5 Message-Digest Algorithm," **RFC 1321**, 1992. [Online]. Available: [\[https://www.rfc-editor.org/rfc/rfc1321\]](https://www.rfc-editor.org/rfc/rfc1321)(<https://www.rfc-editor.org/rfc/rfc1321>)
- [5] P. S. Vora and R. T. Parikh, "Website Defacement Detection Using Hashing Techniques," **International Journal of Computer Science and Information Security**, vol. 17, no. 5, pp. 45–51, 2019.
- [6] M. S. Abiodun, A. J. Oluwadare, and O. O. Oladipo, "Use of Python in Cybersecurity Applications: A Survey," **International Journal of Computer Applications**, vol. 182, no. 5, pp. 25–33, 2021.
- [7] E. Alomari, F. A. Aloul, and K. Al-Azzam, "Website Defacement: Detection, Prevention, and Forensics," **International Journal of Network Security & Its Applications**, vol. 9, no. 6, pp. 55–69, 2017.

[8] T. Ahmed and S. Y. Qureshi, “Real-Time Monitoring of Web Page Integrity Using Python,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 7, pp. 120–128, 2020.

[9] BeautifulSoup Documentation, [Online]

[10] Requests Library Documentation, [Online]. Available: <https://docs.python-requests.org/en/latest/>