applied sciences

MDPI

Article

AV-Teller: Browser Fingerprinting for Client-Side Security Software Identification

Hyeong-Seok Jang [1], Mohsen Ali Alawami [2] and Ki-Woong Park [3,*]

[1] SysCore Laboratory, Sejong University, Seoul 05006, Republic of Korea; rotiple320@gmail.com
[2] Division of Computer Engineering, Hankuk University of Foreign Studies, Yongin 17035, Republic of Korea; mohsencomm@hufs.ac.kr
[3] Department of Information Security, Sejong University, Seoul 05006, Republic of Korea
* Correspondence: woongbak@sejong.ac.kr;

Abstract: The rapid proliferation of digitalization and the growing reliance on internet-based technologies by individuals and organizations have led to a significant escalation in the frequency and sophistication of cyberattacks. As attackers continuously refine their methods to evade conventional defense mechanisms, antivirus solutions, despite their widespread utilization as primary security tools, face increasing challenges in addressing these evolving threats. This study introduces AV-Teller, a novel framework designed for analyzing antivirus behavior through interactions with web browsers. AV-Teller reveals weaknesses in antivirus detection mechanisms by highlighting ways in which web browser interactions may inadvertently expose critical aspects of antivirus operations. The framework provides key insights into the vulnerabilities inherent to these detection processes and their implications for the interplay between antivirus systems and modern web technologies. To assess the efficacy of the AV-Teller in detecting antivirus via web browsers, the framework evaluates three detection scenarios: Document Object Model (DOM) Monitoring-Based Detection, Signature-Based Detection, and Phishing Page-Based Detection. The results revealed performance inconsistencies: 16 products (57%) failed to respond to any tested scenarios, exhibiting deficiencies in threat mitigation capabilities. Of the 12 products (43%) that successfully handled three scenarios, 9 (75%) inadvertently disclosed identifiable antivirus metadata during assessments, thereby enabling attackers to pinpoint specific antivirus solutions and exploit their vulnerabilities. These findings highlight critical gaps in the interaction between antivirus systems and web technologies, exposing systemic flaws in existing security mechanisms. The inadvertent exposure of sensitive antivirus data underscores the necessity for robust data handling protocols, necessitating collaboration between antivirus developers and web technology stakeholders to design secure frameworks. By exposing these risks, the AV-Teller framework elucidates the limitations of current defenses and establishes a foundation for the enhancement of antivirus technologies to address emerging cyber threats effectively.

Keywords: browser fingerprint; antivirus detection; browser security; security evaluation

Academic Editor: Luis Javier García Villalba

Received: 22 March 2025
Revised: 26 April 2025
Accepted: 29 April 2025
Published: 2 May 2025

Citation: Jang, H.-S.; Alawami, M.A.; Park, K.-W. AV-Teller: Browser Fingerprinting for Client-Side Security Software Identification. *Appl. Sci.* **2025**, *15*, 5059. https://doi.org/10.3390/app15095059

1. Introduction

Web browsers are essential components of daily digital interactions, serving as critical interfaces for accessing online information and services. With Google Search facilitating over 80 billion monthly visits [1], adversaries leverage this high traffic to develop sophisticated techniques aimed at circumventing antivirus mechanisms and exploiting systemic vulnerabilities.

Cyberattacks are becoming increasingly complex, with their methods evolving rapidly alongside technological advancements [2–4]. The exploitation of cloud infrastructure and zero-day vulnerabilities has surged in recent years, introducing unprecedented challenges for existing security frameworks [5–7]. Additionally, artificial intelligence (AI) technologies are enabling novel attack vectors, demanding equally advanced and adaptive countermeasures [8–10]. These developments necessitate more sophisticated offensive strategies, which in turn require increasingly complex and responsive defense mechanisms.

The Advanced Persistent Threat (APT) is illustrative of this shift, targeting specific organizations or government entities through prolonged and highly tailored attack strategies [11–14]. A significant objective of APTs is the identification of systems, particularly the detection of security software such as antivirus. The exposure of antivirus information via web browsers represents a critical vulnerability, as attackers can exploit this data without deploying sophisticated detection mechanisms. Specifically, antivirus information identified on the client side, such as product details or behavioral characteristics, can be transmitted externally and verified by third parties. While this capability facilitates legitimate evaluation purposes, it also introduces significant risks, including the potential for misuse by malicious actors. Such exposure exacerbates the risk of targeted attacks, enabling adversaries to develop tailored strategies to bypass antivirus defenses and exploit inherent vulnerabilities.

Browser fingerprinting, a technique that analyzes unique browser attributes to deduce system configurations, has become a significant method for adversaries to detect antivirus software and devise targeted strategies to circumvent detection or exploit vulnerabilities [15–17]. Despite its prevalence, prior studies have not specifically addressed the exposure of antivirus-related information within browser environments. This research represents the first to bridge this gap by employing browser fingerprinting to profile antivirus systems and assess the security risks associated with such disclosures. These insights are essential for developing robust mitigation strategies.

To address these challenges, we introduce AV-Teller, a novel framework that leverages advanced browser fingerprinting techniques to identify antivirus software within browser environments. AV-Teller systematically evaluates antivirus detection mechanisms and assesses associated vulnerabilities. By simulating diverse scenarios and gathering real-time data, the framework facilitates comprehensive analyses of antivirus performance, revealing exploitable weaknesses and offering strategic insights to strengthen security systems.

The remainder of the study is structured as follows: Section 2 outlines the foundational concepts, including antivirus detection strategies, browser-antivirus interactions, DOM monitoring techniques, antivirus evaluation methods, and browser fingerprinting techniques. Section 3 details methodologies for profiling antivirus behavior, encompassing scenario construction and evaluation. Section 4 presents the AV-Teller framework, describing its architectural design and iterative validation process. Section 5 elaborates on the evaluation setup, focusing on design parameters and the testing environment. Section 6 provides an analysis of evaluation, examining antivirus detection capabilities across profiling scenarios. Section 7 discusses insights derived from the findings, potential defense mechanisms, and broader implications for cybersecurity. Finally, Section 8 concludes with a summary of contributions and directions for future research.

2. Background and Related Work

This section provides the necessary background to contextualize the research presented in this paper. It begins by discussing key antivirus detection mechanisms, followed by an examination of the interactions between browsers and antivirus software. Subsequently, it explores the foundational concepts of DOM monitoring and the MutationObserver API, as

well as standardized testing methods used to evaluate antivirus effectiveness. Finally, the section concludes with an overview of browser fingerprinting, highlighting its applications and challenges in the cybersecurity landscape. These topics collectively establish the groundwork for the proposed AV-Teller framework.

2.1. Antivirus Detection Mechanisms

Antivirus software employs diverse detection mechanisms, including signature-based detection, heuristic analysis, behavioral monitoring, and sandboxing, to safeguard systems against malware. These techniques complement one another by addressing distinct aspects of malware detection while confronting unique limitations. Advances such as AI-driven and cloud-integrated methodologies have further augmented the adaptability of antivirus systems to evolving threats.

2.1.1. Signature-Based Detection

Signature-based detection compares file contents against a repository of predefined malware signatures, utilizing hash-based comparisons or string-matching techniques to identify threats. While effective for known malware, its reliance on static patterns renders it inadequate against novel or polymorphic malware, which alter their structure to evade detection [18].

2.1.2. Heuristic and Behavioral Monitoring

Heuristic-based malware detection enhances the identification of previously unknown threats by analyzing anomalous patterns and behaviors that deviate from established baselines. This approach incorporates static analysis, which examines code without execution, and dynamic analysis, which monitors runtime behavior. Behavioral profiling further refines detection capabilities by identifying potentially malicious activities, such as unauthorized file encryption or irregular network interactions. While effective against zero-day threats, these techniques may yield false positives, erroneously classifying benign software as threats.

2.1.3. Sandboxing and Advanced Challenges

Sandboxing establishes a controlled execution environment for analyzing potentially malicious files, effectively mitigating risks associated with sophisticated malware. However, advanced malware frequently employs evasion techniques, such as code obfuscation and polymorphism, to circumvent conventional detection mechanisms [14]. For instance, certain malware variants defer execution until a genuine host system is identified or actively detect virtualized environments to evade sandbox analysis. Polymorphic malware dynamically modifies its structure signature to elude detection, while obfuscation conceals its operational intent, thereby subverting heuristic-based analysis [19].

Modern antivirus systems employ a multifaceted approach to optimize threat detection efficacy. However, the escalating complexity of malware necessitates ongoing advancements and the incorporation of hybrid detection architectures to address emerging vulnerabilities.

2.2. Browser and Antivirus Interactions

Modern antivirus software seamlessly integrates with web browsers to enhance security against threats originating from dynamic web environments. This integration involves monitoring DOM modifications, analyzing webpage components, and inspecting browser activity to detect and mitigate attacks [20]. These mechanisms are essential for identifying anomalous behaviors that could compromise user security.

A key approach is tracking dynamic content behavior within rendered web pages. Antivirus systems analyze browser interactions with web elements, including the detection of unauthorized scripts, the suppression of malicious advertisements, and the identification of suspicious redirections. By systematically observing these interactions, antivirus solutions proactively neutralize embedded threats, thereby strengthening browser-level defenses.

In addition, blacklisting techniques are extensively utilized by antivirus systems to restrict access to known malicious domains and URLs [21]. This method prevents the execution of harmful content within browsers, thereby mitigating phishing attempts and malware dissemination. Recent studies underscore the efficacy of integrating blacklist-based threat detection with browser activity monitoring, enhancing the ability to counter evolving security threats.

Antivirus solutions incorporate security indicators, such as warning notifications or protective overlays, directly into webpages to inform users of potential risks. These dynamically injected elements serve as a proactive defense mechanism, enabling antivirus software to adapt to threats within the browsing environment. This functionality underscore the importance of browser-antivirus interoperability within modern cybersecurity architectures.

2.3. DOM Monitoring and the MutationObserver API

The Document Object Model (DOM) [22] constitutes a standardized representation of the content and structure of a web page, facilitating interactions between web browsers and external systems. Monitoring modifications within the DOM enables the detection of dynamic alterations, including the addition, deletion, or transformation of elements [23], thereby offering insights into web page manipulations performed by legitimate applications or potentially malicious actors.

The MutationObserver API [24] is a JavaScript interface designed for real-time monitoring of DOM changes, offering capabilities to detect attribute modifications, node insertions or deletions, and updates to textual content [25]. Previous research highlights its efficacy in optimizing data collection and enhancing web scraping processes, as it selectively captures relevant modifications without necessitating full DOM traversal. In security contexts, this API has been leveraged to observe JavaScript-driven manipulations, enabling the identification of potential threats such as phishing attempts and unauthorized script injections [26].

This foundational framework for DOM monitoring, coupled with the capabilities of the MutationObserver API, serves as an essential toolset for analyzing dynamic interactions within web environments. These technologies are particularly critical for applications requiring precise, efficient, and accurate detection of real-time changes.

2.4. Antivirus Testing

Antivirus testing is pivotal in assessing the efficacy and dependability of antivirus solutions by replicating diverse cyber threats to evaluate their detection, prevention, and mitigation capabilities. Utilizing standardized test files and controlled experimental setups, these evaluations ensure reproducibility and consistency.

2.4.1. EICAR Test File

The EICAR test file [27], developed by the European Institute for Computer Antivirus Research (EICAR) [28] is a benign code string universally acknowledged as a standard text signature by antivirus software. Upon scanning or execution, the file activates antivirus detection mechanisms, enabling the evaluation of the signature-based detection capabilities of the software in a controlled environment. Though the EICAR file does not emulate real-world malware, it serves as a safe benchmark for verifying antivirus efficacy without risking exposure to actual threats.

2.4.2. AMTSO Phishing Test Pages

The Anti-Malware Testing Standards Organization (AMTSO) [29] offers a standardized framework for antivirus evaluation, featuring test pages that simulate diverse cyber threat scenarios, including phishing attacks, drive-by downloads, and file-based malware delivery. Notably, the AMTSO phishing test page [30] evaluates antivirus responses within web environments, specifically assessing browser-based protection mechanisms. These resources enable researchers to quantitatively measure the effectiveness of antivirus solutions in detecting and mitigating real-world threats in dynamic, browser-driven contexts.

2.4.3. AV-TEST

AV-TEST [31], an independent organization, conducts comprehensive evaluations of antivirus software, assessing malware detection rates, system performance overhead, and usability. Its comparative analyses offer valuable benchmarks for evaluating antivirus effectiveness in real-world scenarios, thereby advancing security technology development.

2.4.4. VirusTotal

VirusTotal [32] is an online platform that aggregates outputs from multiple antivirus engines to evaluate files and URLs for malicious content. It enables efficient threat analysis and facilitates verification of antivirus detection efficacy across diverse engines. Additionally, VirusTotal serves as a practical tool for end-users and a research resource for investigating antivirus performance and malware identification.

2.5. Browser Fingerprinting

Browser fingerprinting is a technique employed to uniquely identify a browser by collecting specific attributes from its environment. Previous research has predominantly focused on gathering information such as browser version, system time zone, preferred language settings, supported video and audio codecs, installed system fonts, browser configurations, and device display parameters including resolution and dimensions [33,34]. Such information has been leveraged extensively for applications such as targeted advertising and personalized services by verifying user devices and adapting content to align with individual preferences [35–37].

While the majority of studies in browser fingerprinting [33,36] have focused on user or device identification through static and dynamic browser characteristics, limited attention has been devoted to investigating whether browser-level observations can infer the presence of security software, such as antivirus products.

AV-Teller introduces a novel dimension to system profiling by detecting antivirus-specific artifacts. These artifacts, including DOM manipulations and injected elements, emerge during browser interactions and provide passive, reproducible signals for remotely identifying antivirus presence. Notably, this detection approach circumvent the need for code execution, privileged system access, or user cooperation.

This approach diverges from conventional fingerprinting research both in its aim and technical framework. Rather than focusing on profiling users or devices, AV-Teller emphasizes the interaction of browser with system-level security mechanisms, thereby broadening fingerprinting methodologies with a security-aware perspective.

Furthermore, our method extends prior research by seamlessly integrating with established fingerprinting pipelines. AV-Teller operates concurrently with conventional identification methods, enabling a comprehensive characteristics of client environment. This integration facilitates the development of multi-faceted system profiles that encompass both usability and security features.

3. Strategies and Tactics for Profiling Antivirus Behavior

This section outlines the methodology devised for profiling antivirus software via web browsers and details the procedure for construct scenarios. The conceptualization and design of the antivirus profiling scenarios and methodologies presented in this study were exclusively undertaken by the author. Four distinct approaches were utilized to develop these scenarios, as described below.

3.1. Approaches for Constructing Antivirus Profiling Scenarios

Antivirus profiling fundamentally entails the specification of detection conditions and scenario construction to evaluate antivirus responses within a browser environment. The detection conditions were defined through four primary approaches:

- Exploration of Configuration Parameters: The diverse configuration parameters of antivirus solutions were systematically analyzed to assess their influence on detection mechanisms. For instance, comparative evaluation were conducted to analyze behavioral differences when real-time protection was toggled and to monitor system responses to altered advanced settings.
- Reverse Engineering: Reverse engineering techniques were employed to analyze the internal workings of antivirus software. This approach facilitated deeper comprehension into the algorithms used by detection engines under specific conditions and their interactions with web browsers. However, most antivirus software incorporates robust anti-reversing techniques, making the analysis both challenging and time-consuming. Particularly, mechanisms such as code obfuscation and anti-debugging significantly increased the effort required to understand the detection logic and operational processes.
- Analysis of Patent Information: Given that antivirus detection technologies are frequently safeguarded by patents, an examination of patent documentation provided critical insights into the theoretical foundations underlying specific detection methodologies. These insights were instrumental in identifying the distinctive detection mechanisms employed by various antivirus solutions and in formulating corresponding scenarios.
- Observation of Browser Interactions: Direct observation of interactions between web browsers and antivirus software enable the identification of key detection parameters. This involved monitoring behaviors, such as antivirus interference with loaded web pages.

Figure 1 illustrates the four approaches described above employed to establish detection conditions for antivirus profiling: configuration parameter exploration, reverse engineering, patent analysis, and browser interaction observation. These approaches informed the development of three primary detection scenarios, forming the foundation for evaluating antivirus systems responses under specific conditions.

3.2. Antivirus Profiling Scenarios

Three key scenarios were designed to evaluate the detection performance of the antivirus software in web browser environments. Each scenario was constructed based on specific detection conditions and aimed to analyze the antivirus detection mechanisms in detail.

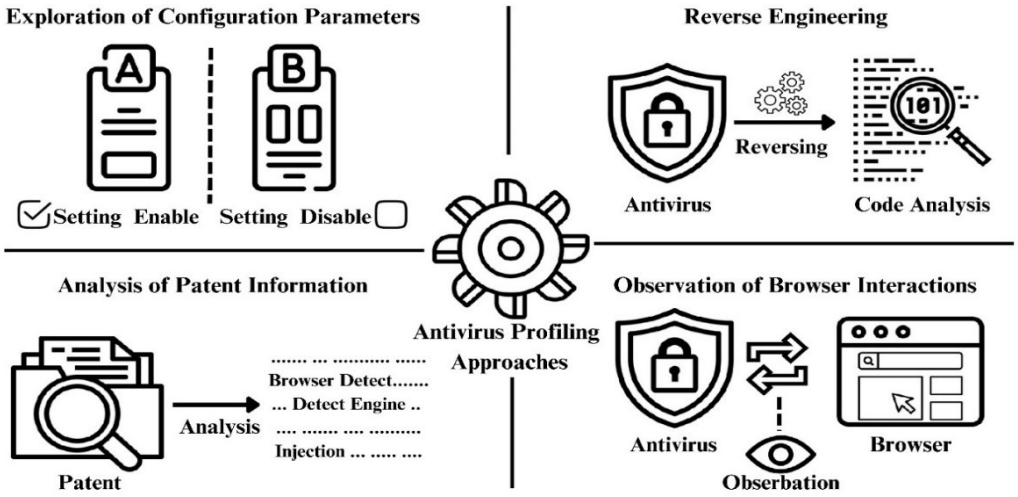

Figure 1. Conceptual summary of antivirus profiling approaches.

3.2.1. Scenario 1: DOM Monitoring-Based Detection

The Document Object Model (DOM) is a standardized hierarchical representation of web page content and structure, encompassing HTML elements (e.g., div, p, and span), attributes, and event handlers. It serves as the primary interface for interactions between the web browser and user, positioning the DOM as a critical target for antivirus solutions aiming to detect and mitigate suspicious elements. Antivirus software often modifies the DOM by altering its structure or injecting additional elements to identify and neutralize malicious code. For instance, common actions include inserting security alert banners or removing specific nodes in the DOM to block harmful scripts, providing insights into the interplay between browsers and antivirus software. Scenario 1 employs the Web API MutationObserver for real-time monitoring of DOM modifications, including element addition/removal, attribute value alterations, and text content updates. By observing these changes, Scenario 1 identifies antivirus characteristics through real-time tracking of DOM modifications.

Figure 2 illustrates the functionality of the MutationObserver API for real-time monitoring of changes to the DOM structure. The diagram depicts the hierarchical structure of the DOM, where black lines denote parent-child relationships between elements, such as HTML, body, and img. These connections represent the document structure, linking parent nodes to their corresponding child nodes. Arrows within the diagram categorize distinct types of DOM alterations: blue arrows represent additions (e.g., the insertion of new elements into the DOM), red arrows signify deletions (e.g., the removal of elements or nodes), and green arrows indicate modifications (e.g., updates to text content or attribute values). The red box highlights interactions initiated by antivirus software within the DOM, such as the addition of security-related elements (e.g., banners or warning messages) aimed at mitigating potential threats. The MutationObserver captures these interactions, enabling comprehensive analysis of antivirus-induced modifications and their behavioral implications. This approach facilitates the identification of antivirus activities and their real-time impact on the DOM structure.

To validate the source of DOM modifications, we analyzed their consistency across multiple test environments. These changes—such as injected warning elements, vendor-specific script tags, and uniquely styled DOM nodes—were consistently replicated in successive test runs and exhibited behavior specific to individual antivirus solutions. Although we did not conduct a dedicated control experiment with isolated browser extensions due to time constraints, we confirmed that such mutations did not appear in clean envi-

ronments or when common extensions were enabled. These results strongly indicate that the observed modifications originated from antivirus interventions rather than regular browser behavior.

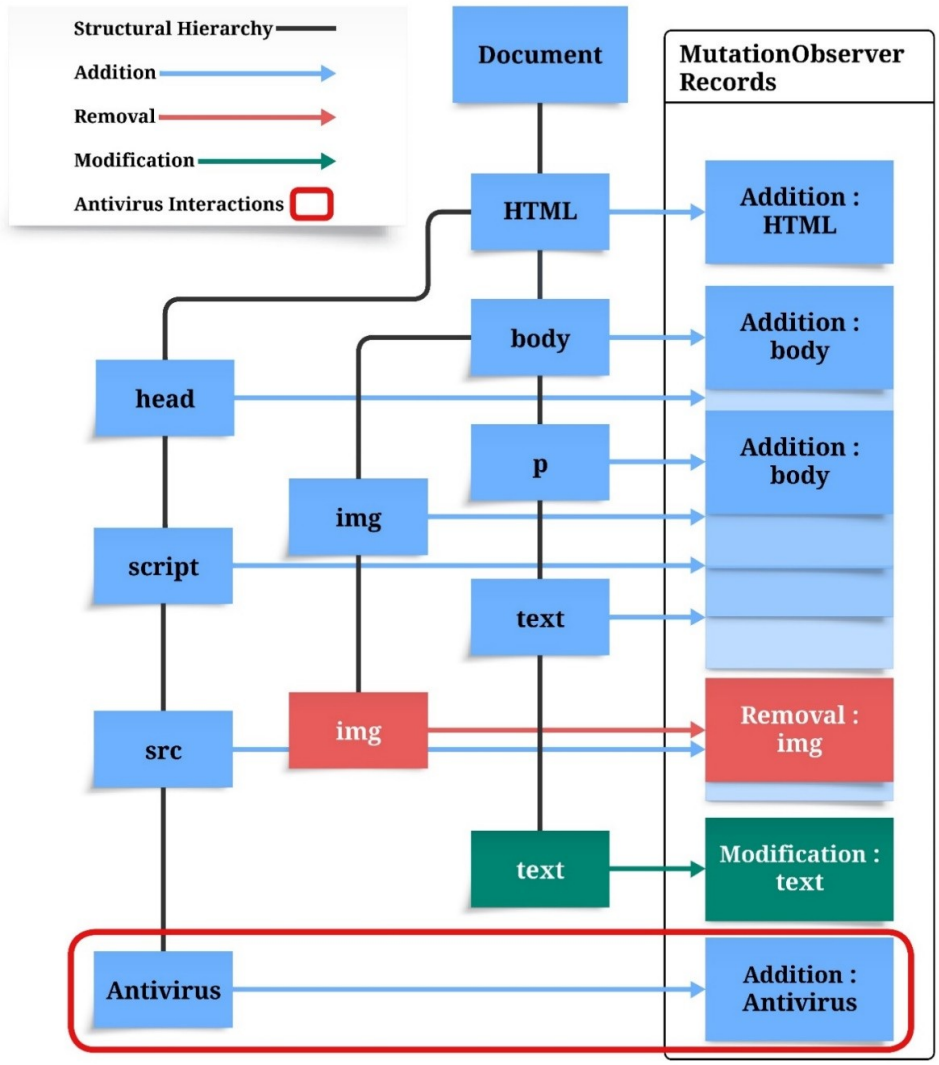

Figure 2. DOM monitoring using MutationObserver API for antivirus profiling.

3.2.2. Scenario 2: Signature-Based Detection

Antivirus software commonly employs signature-based detection mechanisms to identify malicious code. These mechanisms rely on predefined patterns, or "signatures", of malicious code, enabling detection engines to identify threats efficiently and accurately.

Scenario 2 profiles antivirus software by leveraging signature-based detection capabilities. To trigger detection, the EICAR signature, a widely recognized test string used to verify antiviral functionality, was embedded in the webpage content. When encountering an EICAR string, some antiviral solutions classify it as a malicious signature, blocking access to the page or displaying warning messages. This behavior provides valuable insights into the detection characteristics of specific antiviral software packages.

Figure 3 illustrates the signature-based detection process employed by antivirus software. When a webpage containing an EICAR test file is accessed through a browser, the antivirus system scans the file by comparing it with its signature database. After matching the EICAR signature, the antivirus software identifies the webpage as malicious and blocks

access, thereby providing valuable insights into the detection mechanisms utilized by the antivirus engine.

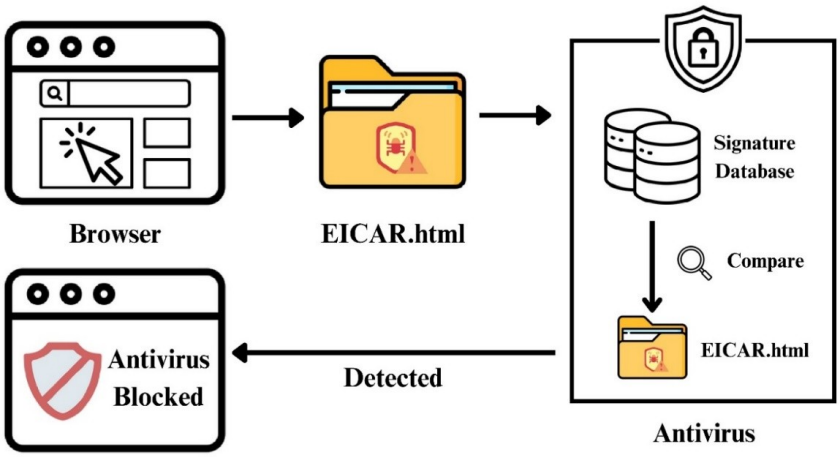

Figure 3. Signature-Based detection process.

3.2.3. Scenario 3: Phishing Page-Based Detection

Phishing pages are malicious websites engineered to deceive users into divulging sensitive personal or financial information. These pages represent a critical cybersecurity threat, making their detection a key focus for modern antivirus solutions. Given the sophistication of phishing attacks, which often closely mimic legitimate websites, the detection performance of antivirus software plays a vital role in ensuring cybersecurity. Scenario 3 evaluates antivirus software by analyzing their phishing detection mechanisms. To maintain a safe experimental environment, test phishing pages provided by AMTSO were used instead of real phishing URLs. This approach enables the evaluation of antivirus capabilities to identify phishing threats while ensuring the safety of the testing setup.

Figure 4 depicts the process of phishing-page detection using antivirus software. When a browser accesses a test phishing page provided by AMTSO, the antivirus system inspects the URL or analyzes the webpage content to identify the phishing characteristics. Upon detecting a phishing threat, the antivirus blocks access to the page or displays a warning, thereby enabling the evaluation of its phishing detection capabilities in a controlled environment.

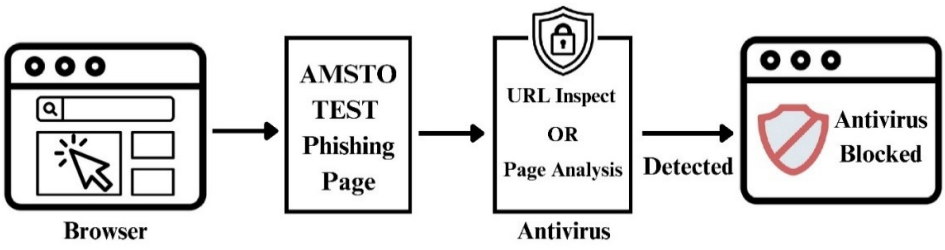

Figure 4. Phishing Page-Based detection process.

The three scenarios were selected to represent distinct and practical antivirus detection categories: (1) behavioral detection through DOM observation, (2) static signature-based detection using the EICAR test file, and (3) heuristic or URL-based detection via phishing simulation. These scenarios were selected for their reproducibility, clarity of outcome, and alignment with the detection mechanisms commonly supported by modern antiviral solutions. However, this selection does not encompass the full spectrum of real-world threats. The EICAR test file is a synthetic test artifact that does not simulate the complexity of advanced malware. Furthermore, detection techniques, such as memory-resident analysis, encrypted

traffic inspection, and cloud-assisted threat classification, were not considered in the current scope. Nevertheless, the selected scenarios provide a meaningful baseline for evaluating antivirus behavior in a controlled setting. These findings may serve as a foundation for future studies involving more sophisticated and evasive threat models. Using these three scenarios, we established a comprehensive framework for analyzing antiviral detection mechanisms and identifying vulnerabilities in browser environments. Each scenario includes experimental validation under specific detection conditions, with the goal of evaluating the performance and suggesting potential improvements. The following section provides a detailed overview of the design and implementation of the AVTeller framework that was employed to execute the scenarios and analyze the data collected from them.

4. Implementation of the AV-Teller Framework

This section outlines the design and implementation of the AV-Teller framework. The focus was on the validation method, architectural design of the AV-Teller framework, and key features that facilitated the analysis of antiviral detection mechanisms.

4.1. Iterative Validation Method

We propose an iterative validation framework to analyze and verify antivirus detection mechanisms. The method involves defining detection conditions, formulating evaluation scenarios based on these conditions, validating scenarios through experimental procedures, and leveraging experimental feedback to iteratively refine detection conditions. This cyclical approach mitigates the constraints of predefined detection conditions and supports experimental refinement by adapting conditions to account for newly identified vulnerabilities. Figure 5 illustrates the validation process, encompassing antivirus analysis, detection conditions formulation, experimental validation, and the feedback loop for evaluating detection systems. The framework involves the following components:

1. Antivirus Analysis: Initial data for antivirus profiling is collected through four key methods: exploring configuration parameters, conducting reverse engineering, analyzing patent information, and observing browser interactions.
2. Scenario Establishment: The detection conditions derived from this analysis are applied to construct specific scenarios for evaluating antivirus behavior.
3. AV-Teller Framework Validation: Constructed scenarios are validated using the AV-Teller Framework.
4. Feedback Loop: Validation outcomes inform further refinement of detection conditions and scenario development through iterative feedback.

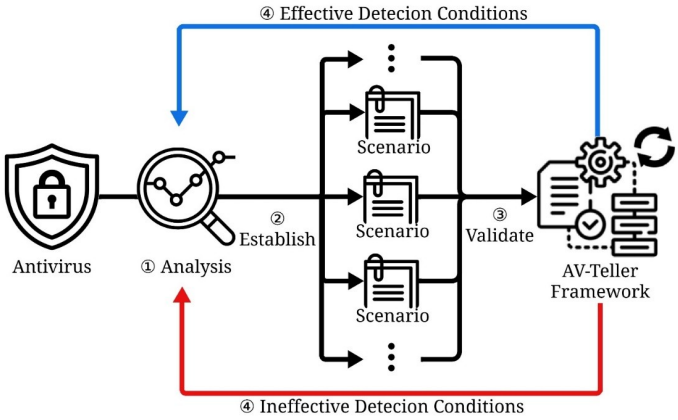

Figure 5. A flow diagram for detecting antivirus.

4.2. Design of the AV-Teller Framework

Figure 6 illustrates an overview of the framework, detailing interactions among core modules and the associated data flow. To enhance comprehensibility, Table 1 summarizes the development environment and toolset utilized in implementing the AV-Teller framework, establishing the technological foundation essential for its architecture and evaluation.

Table 1. Development environment and tools for AV-Teller framework.

Category	Details
Development Environment	Windows 11 on Intel Core i9-14900k, 64 GB RAM
Programming Languages	Python (v3.10), PowerShell (v5.1), JavaScript (ES2024), HTML (HTML5)
Libraries/Frameworks	MutationObserver API, Hyper-V API, Flask, Selenium
Testing Platform	Hyper-V on Windows 10 with Google Chrome (v131.0), Firefox (v132.0), Microsoft Edge (v131.0)
Database	SQLite (v3.47.1)
Tools/IDEs	Visual Studio Code (v1.85), Git (v2.44.0)
Experiment Configuration	Default antivirus settings, browser without extensions

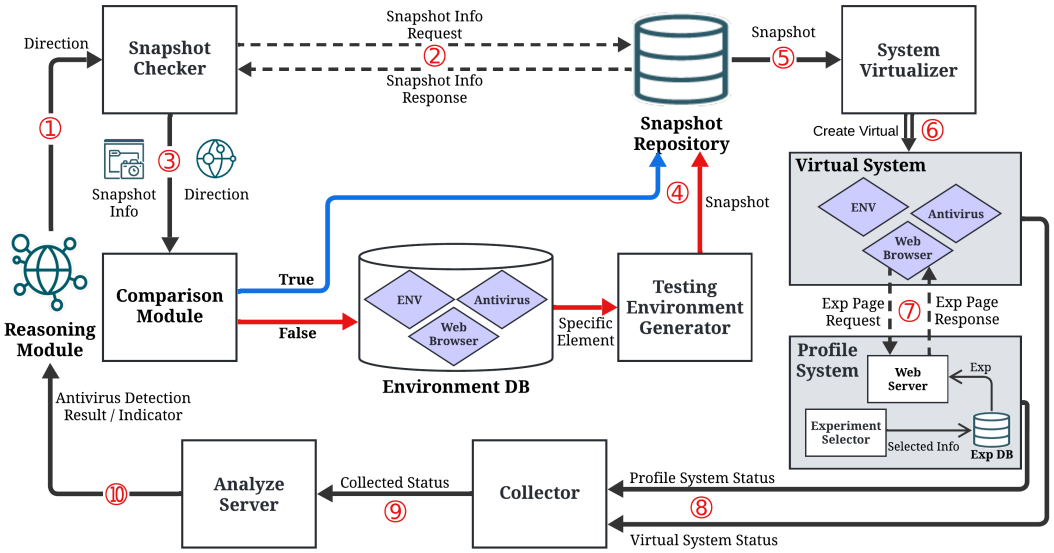

Figure 6. An overview of the architecture for AV-Teller Framework.

The subsequent sections outline the designed and implementation of these components within the specified environment:

- Reasoning Module: Initializes the process by accepting user input and analyzing antivirus detection results from the Analyze Server. It determines a new experimental trajectory, communicates it to the Snapshot Checker, and iterates the process.
- Snapshot Checker: Retrieves snapshot data from the Snapshot Repository and transmits it to the Comparison Module along the specified analytical pathway.
- Comparison Module: Evaluates received data against existing snapshots in the repository. If no correlation is detected, it queries the Environment DB to construct a new snapshot.
- Environment DB: Stores configurations parameters for antivirus software, web browsers, and experimental environments. It provides essential elements for the Testing Environment Generator.
- Testing Environment Generator: Constructs new system snapshots utilizing elements from the Environment DB, which are subsequently stored in the Snapshot Repository.

- Snapshot Repository: Maintains snapshots with unique configurations of antivirus software and web browsers, enabling their retrieval for the Snapshot Checker when required.
- System Virtualizer: Facilitates the virtualization of system environments through snapshots from the repository.
- Virtual System: Executes experimental procedures by interfacing with the Profile Server via a web browser configured with various environmental settings
- Profile Server: Emulates test scenarios, sourced from the Experiment DB, through its web server while interacting with the browser of the virtual system during experimentation.
- Collector: Collects data on system states and experimental results, transmitting them to the Analyze Server for further processing.
- Analyze Server: Processes experimental data to derive detection outcomes and performance metrics, subsequently relaying results to the Reasoning Module.

5. Evaluation

The AV-Teller framework was employed to evaluate the efficacy of antivirus detection mechanism. By leveraging system snapshots, it reconstructed diverse configurations, comprising various antivirus products combined with different browser settings. The evaluation prioritized simulating a representative and widely adopted environment, focusing on prevalent usage scenarios. Thus, antivirus settings remained at their default configurations, and browser extensions were excluded to eliminate potential biases introduced by atypical setup. This approach provided a consistent baseline for analysis. Integral to the AV-Teller framework, the Collector module systematically recorded detection events and system responses during experimentation. The collected data were processed by the Analyze Server, which generated critical performance metrics, including detection rates and false positives. Iterative analyses facilitated the identification of system vulnerabilities and the refinement of detection scenarios.

To ensure the validity of the experimental setup, real-world usage patterns were considered, as supported by data from StatCounter [38], a web analytics service tracking global browser and operating system market shares. As of October 2024, StatCounter reported Chrome's dominance in the global desktop browser market, with a 65.25% share, establishing it as the most widely utilized browser. Similarly, Windows accounted for 73.39% of the desktop operating system market, with Windows 10 comprising 60.95% of that share. These statistics substantiate the selection of Chrome and Windows 10 as primary platforms for evaluation, aligning the experimental setup with prevailing user environments.

For antivirus evaluation, we selected 28 products certified by AV-TEST and registered on VirusTotal. AV-TEST is an independent organization that evaluates and certifies antivirus products based on criteria such as detection performance and usability. VirusTotal aggregates results from multiple antivirus engines, reflecting their market usage and relevance. These criteria ensured the selection of widely used and credible antivirus products.

6. Results

This section reports the results of evaluating antivirus detection mechanisms within the AV-Teller Framework, utilizing the evaluation setup detailed in Section 5. The assessments encompassed three detection scenarios: DOM Monitoring-Based Detection, Signature-Based Detection, and Phishing Page-Based Detection. The findings demonstrated varying levels of efficacy across the evaluated antivirus solutions.

The evaluation outcomes were categorized into three distinct classes: Antivirus Identification, Protected Anonymous, and Antivirus Undetected. Antivirus Identification denotes instances where the antivirus successfully executes its protective function—such as threat

blocking or system activity indicative of antivirus operations—yet inadvertently discloses identifiable information that adversaries could exploit.

Protected Anonymous represents the optimal outcome, wherein the antivirus effectively mitigates threats or interacts with the system while maintaining anonymity, thereby minimizing exposure to targeted attacks. Antivirus Undetected corresponds to scenarios where no discernible antivirus activity was observed, rendering the system unprotected and vulnerable against potential threats. Table 2 provides a comprehensive summary of these categories, including their definitions, percentages, and the number of antivirus products observed within each group.

Table 2. Detection results overview.

Category	Definition	Rate	Count
Antivirus Identification	The antivirus exhibited behaviors, such as threat blocking or activity execution, that revealed identifiable information.	32%	9/28
Protected Anonymous	The antivirus successfully performed its functions without revealing its identity.	11%	3/28
Antivirus Undetected	No antivirus-related activity was observed, leaving the system unprotected and susceptible to threats.	57%	16/28

The evaluation revealed that 32% (9 products) of the tested antivirus were categorized under Antivirus Identification, 11% (3 products) achieved Protected Anonymous, and 57% (16 products) fell into the Antivirus Undetected category. While Antivirus Identification indicates the capacity to block threats, it remains insufficient due to the exposure of the antivirus system to potential exploitation. The Antivirus Undetected outcomes category presents the greatest concern, leaving users entirely vulnerable to threats and emphasizing critical system vulnerabilities. Conversely, Protected Anonymous represents an optimal balance, combining effective threat mitigation with anonymity, which is crucial for preventing attackers from leveraging antivirus-specific weaknesses. Table 3 provides further insights into the individual performance of antivirus products across three detection methods: Sc. 1 DOM Monitoring-Based Detection, Sc. 2 Signature-Based Detection, and Sc. 3 Phishing Page-Based Detection, offering a comprehensive view of their detection capabilities. These findings underscore the varied efficacy of antivirus solution across distinct detection methodologies.

To ensure the reliability of our results, each experiment was conducted 100 times per antivirus product across all three scenarios, producing consistent detection outcomes with negligible variance. Additionally, the experiments also included executions within a no-antivirus (AV) baseline environment to verify that detection artifacts—such as DOM modifications, signature alerts, or phishing blocks—did not originate from browser behavior alone. This confirms that all observed effects stemmed from antivirus-specific responses.

Antivirus products were classified based on their responses across the three scenarios. Products exhibiting an "**O**" (Antivirus Identification) in any scenario were categorized as Antivirus Identification, whereas products demonstrating a "▲" (Protected Anonymous) in any scenario were classified as Protected Anonymous. Since detection in even a single scenario reflects behavior corresponding to the respective classification, this hierarchical approach prioritizes the most significant behavior—either identification or anonymous protection—in the final categorization.

To further illustrate the overall detection patterns across the three AV-Teller scenarios a summary chart is presented in Figure 7.

Table 3. A test results for three evaluations from Chrome web browser.

Targeted Antivirus	Sc. 1	Sc. 2	Sc. 3	Result
Kaspersky Premium (v21.19)	**O**	**O**	**O**	Antivirus Identification
Avast Free Antivirus (v24.10)	▲	▲	X	Protected Anonymous
AVG Internet Security (v21.10)	▲	▲	X	Protected Anonymous
ESET Home Security Premium (v18.0.12.0)	▲	**O**	X	Antivirus Identification
McAfee Total Protection (v1.24.165)	X	X	X	Antivirus Undetected
Avira Free Security (v1.1.105)	X	X	X	Antivirus Undetected
Bitdefender Total Security (v7.9.17.458)	**O**	**O**	X	Antivirus Identification
360 Total Security (v11.0.0.1163)	X	X	X	Antivirus Undetected
F-Secure Total (v19.7)	X	X	X	Antivirus Undetected
K7 Total Security (v16.0.0.1241)	X	**O**	X	Antivirus Identification
Ahnlab V3 365 Clinic (v4.11.1.427)	X	X	X	Antivirus Undetected
G Data Internet Security (v25.5.18)	X	**O**	X	Antivirus Identification
webroot Internet Security Plus (v9.0.38.39)	X	X	X	Antivirus Undetected
Sophos Home Premium (v2023.2.2.2)	X	**O**	X	Antivirus Identification
Vipre Advanced Security (v12.0.2.220)	X	**O**	X	Antivirus Identification
ALYac (v2.5.14)	X	X	X	Antivirus Undetected
Arcabit Internet Security (v2024.11.17)	X	**O**	**O**	Antivirus Identification
ClamAV (v1.4.1)	X	X	X	Antivirus Undetected
Emsisoft Anti-Malware Home (v2024.11.0.12611)	X	X	X	Antivirus Undetected
Dr.Web Security Space (v12.06.26.10170)	X	**O**	X	Antivirus Identification
CMC Antivirus (v2.3.1)	X	X	X	Antivirus Undetected
Bkav Internet Security Pro(v8.2.1)	X	X	X	Antivirus Undetected
Escan Total Security Suite (v22.0.1400.2800)	X	X	X	Antivirus Undetected
Virobot Security 1.0 (v1.1.3.6)	X	X	X	Antivirus Undetected
Huorong Internet Security (v5.0.76.3)	X	▲	X	Protected Anonymous
Comodo Antivirus (v12.3.3.8152)	X	X	X	Antivirus Undetected
Malwarebytes Premium Security (v5.2.1.144)	X	X	X	Antivirus Undetected
TACHYON Internet Security 6.0 (6.0.5.2)	X	X	X	Antivirus Undetected

All evaluations were conducted on 17 November 2024; Sc. 1: DOM-based, Sc. 2: Signature-based, Sc. 3: Phishing emulation; **O**: Antivirus Identification, ▲: Protected Anonymous, X: Antivirus Undetected.

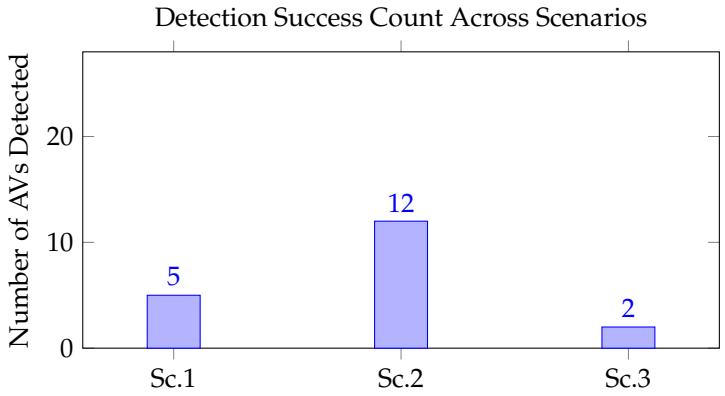

Figure 7. Summary of detection counts for Sc.1 (DOM Monitoring), Sc.2 (Signature-Based), and Sc.3 (Phishing Page-Based).

7. Discussion

The evaluation demonstrated diverse performance outcomes among antivirus systems, with notable strengths and vulnerabilities. Certain systems successfully mitigated threats, while others exhibited substantial inconsistencies, particularly in phishing detection and DOM-based attack scenarios. These results emphasize the necessity for further investigation into broader security implications and the development of targeted solutions to address identified gaps.

7.1. Lessons Learned

The AV-Teller framework proved effective in evaluating antivirus detection mechanisms, offering flexibility for experimental setups across various browser and antivirus configurations. Its adaptability facilitates testing under diverse system environments and addressing emerging security challenges. Enhanced real-time data collection and analysis capabilities improved the detection of vulnerabilities and refinement of detection scenarios.

Notably, minimal browser interactions revealed insight into antivirus system behaviors, exposing inherent weaknesses in client-side security and underscoring the necessity for more robust detection mechanisms to mitigate information leakage risks. Additionally, the observed inconsistency in antivirus performance across scenarios, especially in phishing detection and DOM-based monitoring, indicates current algorithmic limitations in addressing diverse threat vectors. These findings highlight the need for advancements in antivirus algorithms to ensure resilience across a broader spectrum of attack scenarios.

The scalability of the AV-Teller framework offers substantial potential for further research. Although this study primarily addressed desktop environments, the methods are adaptable to mobile and cloud-based systems. This adaptability establishes AV-Teller as a robust foundation for investigating detection mechanisms across diverse emerging technologies, ensuring sustained relevance in future security evaluations.

Variations in antivirus performance observed during testing likely stem from inherent differences in detection capabilities and vendor-specific strategic priorities. AV-Teller evaluates three distinct detection vectors—behavioral (DOM manipulation), static (EICAR signature), and heuristic or URL-based (phishing emulation). While certain antivirus solutions demonstrated consistent efficacy across all vectors, others exhibited gaps in specific scenarios. For instance, products without real-time DOM analysis failed to detect Scenario 1, and limited URL filtering hindered the detection of phishing attempts. Such discrepancies reflect architectural and policy-driven differences rather than inherent deficiencies.

7.2. Scenario Limitations and Generalizability

While AV-Teller evaluates three representative detection mechanisms—behavioral (DOM monitoring), signature-based (EICAR test), and heuristic/URL-based (phishing emulation)—these scenarios do not fully generalize to the broad spectrum of real-world threats. For instance, the EICAR file employed in the signature-based scenario is a synthetic test string and may not reflect detection behaviors against more complex or evasive malware.

Additionally, advanced evasion techniques such as memory-resident malware analysis, cloud-based threat analysis, encrypted payload inspection, and delayed execution bypasses remain outside the scope of the study. Consequently, antivirus responses observed in our study may vary under more complex attack vectors.

Nevertheless, the selected scenarios provide a reproducible framework for assessing antivirus performance under browser-based attack conditions and serve as a foundation for future research that integrates broader threat models to improve the comprehensiveness and robustness of antivirus detection profiling.

To ensure statistical rigor, each experiment was executed 100 times for all antivirus products across all scenarios. Detection behavior remained consistent, allowing for aggregate results. All scenarios were conducted in a controlled, antivirus-free environment, ensuring that observed effects arose solely from antivirus intervention. Importantly, no false-positive behaviors (e.g., DOM mutations or phishing alerts) were identified.

7.3. Possible Defense Strategies

Addressing the vulnerabilities identified in this study and enhancing antivirus detection mechanisms necessitate coordinated efforts between browser developers and antivirus vendors. Collaborative frameworks are pivotal for mitigating risks and fortifying the resilience of security systems.

Standardized threat response protocols, such as unified signals for detection events (e.g., HTTP response codes or DOM interaction logs), are crucial to prevent attackers from exploiting unique patterns to target specific antivirus solutions. Strengthening browser-level security to restrict external web pages from accessing antivirus-related data is essential for reducing fingerprinting attack vectors. These measures protect sensitive user data and minimize antivirus systems' exposure to external threats.

Unified threat detection approaches that integrate antivirus and browser systems can further mitigate vulnerabilities. Collaborative frameworks, where browsers and antivirus systems share detection insights while maintaining user anonymity, can strengthen overall security without compromising system integrity. Such coordination can also prevent attackers from exploiting predictable response patterns, thereby improving user privacy and system robustness. By adopting these strategies, vulnerabilities in antivirus detection mechanisms can be addressed, ensuring better protection against evolving cyber threats. These improvements will strengthen individual systems and contribute to a more secure and resilient digital ecosystem.

7.4. Portability, Threat Model, and Disclosure

This study specifically examines Windows-based environments due to their ubiquity and the robust testing interfaces provided by leading antivirus solutions. However, antivirus behavior exhibits considerable variability across other platforms, including macOS, Linux, and mobile operating systems. Extending the capabilities of AV-Teller to these platforms represent a significant avenue for future research. Each platform introduces distinct technical challenges, including variations in browser architecture, JavaScript API compatibility, and antivirus integration mechanisms.

For instance, in Android environments, integration with WebView could enable the monitoring of DOM-level changes analogous to traditional browsers, contingent on the support of WebView for JavaScript-based instrumentation. On cloud-based platforms such as AWS WorkSpaces or Azure Virtual Desktop, AV-Teller could function similarly to conventional desktop browser, provided that browser access and network stack behavior remain consistent. However, cloud-specific constraints, including shared kernel execution, session isolation, and endpoint security policies, may impact observability and detection mechanisms.

Responsible disclosure remains a critical consideration when publishing research that could reveal unintended behaviors in security products. Although no direct disclosure to antivirus vendors occurred during the preparation of this manuscript, we aim to disseminate our findings in follow-up work to foster transparency and improve browser-antivirus interactions.

8. Conclusions

In this study, we introduced the AV-Teller framework to evaluate the information exposure risks posed by antivirus during browser interactions, employing three detection scenarios: DOM Monitoring-Based Detection, Signature-Based Detection, and Phishing Page-Based Detection. Our analysis identified significant inconsistencies in antivirus performance, with numerous solutions failing to detect or block threats effectively. Among the 28 antivirus products examined, 16 (57%) failed to detect any threats. Of the remaining 12 products that successfully mitigated threats, 9 (75%) disclosed their identity, exposing a critical vulnerability in preserving anonymity. While some systems demonstrated an optimal balance between threat mitigation and anonymity, others exhibited significant shortcomings, particularly in DOM Monitoring-Based and Phishing Page-Based detection scenarios. These observations underscore the necessity for enhancing detection algorithms and fortifying client-side security mechanisms. Although our findings are based on a representative sample of 28 antivirus products, we acknowledge potential limitations in extrapolating these results across the broader antivirus ecosystem.

Author Contributions: Conceptualization, methodology, experiments, software, writing—original draft preparation, H.-S.J.; validation, visualization, investigation, writing—review and editing, M.A.A.; supervision, project administration, funding acquisition, K.-W.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) (Project No. RS-2023-00228996, 30%; RS-2024-00438551, 30%, IITP-2025-RS-2021-II211816, 10%), and the National Research Foundation of Korea (NRF) grant funded by the Korean Government (Project No. RS-2023-00208460, 30%).

Data Availability Statement: The dataset will be available upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Statista. Worldwide Visits to Google.com from October 2023 to March 2024. 2025. Available online: https://www.statista.com/statistics/268252/web-visitor-traffic-to-googlecom (accessed on 12 March 2025).
2. Mallick, M.A.I.; Nath, R. Navigating the cyber security landscape: A comprehensive review of cyber-attacks, emerging trends, and recent developments. *World Sci. News* **2024**, *190*, 1–69.
3. Murali, R.J.; K, B.S.S.S.; Raj, S.O.N.N. An Extensive Examination of Cyber Attacks and Cyber Security, Encompassing Recent Advancements and Emerging Trends. In Proceedings of the 2024 Ninth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Chennai, India, 4–5 April 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 1–5.
4. Bendovschi, A. Cyber-attacks–trends, patterns and security countermeasures. *Procedia Econ. Financ.* **2015**, *28*, 24–31. [CrossRef]
5. Zengeni, I.P.; Zolkipli, M.f. Zero-Day Exploits and Vulnerability Management. *Borneo Int. J.* **2024**, *7*, 26–33.
6. Sarkar, S.; Choudhary, G.; Shandilya, S.K.; Hussain, A.; Kim, H. Security of zero trust networks in cloud computing: A comparative review. *Sustainability* **2022**, *14*, 11213. [CrossRef]
7. Swathy Akshaya, M.; Padmavathi, G. Zero-day attack path identification using probabilistic and graph approach based back propagation neural network in cloud. *Math. Stat. Eng. Appl.* **2022**, *71*, 1091–1106.
8. Gao, Y. Cyber Attacks and Defense: AI-Driven Approaches and Techniques. *Acad. J. Comput. Inf. Sci.* **2024**, *7*, 41–46.
9. Guembe, B.; Azeta, A.; Misra, S.; Osamor, V.C.; Fernandez-Sanz, L.; Pospelova, V. The emerging threat of ai-driven cyber attacks: A review. *Appl. Artif. Intell.* **2022**, *36*, 2037254. [CrossRef]
10. Yamin, M.M.; Ullah, M.; Ullah, H.; Katt, B. Weaponized AI for cyber attacks. *J. Inf. Secur. Appl.* **2021**, *57*, 102722. [CrossRef]
11. Krishnapriya, S.; Singh, S. A Comprehensive Survey on Advanced Persistent Threat (APT) Detection Techniques. *Comput. Mater. Contin.* **2024**, *80*, 2675–2719. [CrossRef]
12. Sharma, A.; Gupta, B.B.; Singh, A.K.; Saraswat, V. Advanced persistent threats (apt): Evolution, anatomy, attribution and countermeasures. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 9355–9381. [CrossRef]
13. Fahad, M.; Airf, H.; Kumar, A.; Hussain, H.K. Securing against apts: Advancements in detection and mitigation. *BIN Bull. Inform.* **2023**, *1*, 59–70.

14. Sharma, A.; Gupta, B.B.; Singh, A.K.; Saraswat, V. Orchestration of APT malware evasive manoeuvers employed for eluding anti-virus and sandbox defense. *Comput. Secur.* **2022**, *115*, 102627. [CrossRef]
15. Radivojevic, K.; Clark, N.; Klempay, A.; Brenner, P. Defending novice user privacy: An evaluation of default web browser configurations. *Comput. Secur.* **2024**, *140*, 103784. [CrossRef]
16. Sim, K.; Heo, H.; Cho, H. Combating Web Tracking: Analyzing Web Tracking Technologies for User Privacy. *Future Internet* **2024**, *16*, 363. [CrossRef]
17. Lin, X.; Araujo, F.; Taylor, T.; Jang, J.; Polakis, J. Fashion faux pas: Implicit stylistic fingerprints for bypassing browsers' anti-fingerprinting defenses. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–25 May 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 987–1004.
18. Botacin, M.; Alves, M.Z.; Oliveira, D.; Grégio, A. HEAVEN: A Hardware-Enhanced AntiVirus ENgine to accelerate real-time, signature-based malware detection. *Expert Syst. Appl.* **2022**, *201*, 117083. [CrossRef]
19. Parveen, K. Advanced techniques of malware evasion and bypass in the age of antivirus. *Int. J. Electron. Crime Investig.* **2024**, *8*, 25–40.
20. Shahriar, H.; Weldemariam, K.; Zulkernine, M.; Lutellier, T. Effective detection of vulnerable and malicious browser extensions. *Comput. Secur.* **2014**, *47*, 66–84. [CrossRef]
21. Tsirantonakis, G.; Ilia, P.; Ioannidis, S.; Athanasopoulos, E.; Polychronakis, M. A Large-scale Analysis of Content Modification by Open HTTP Proxies. In Proceedings of the NDSS, San Diego, CA, USA, 18–21 February 2018.
22. MDN Web Docs. Document Object Model (DOM). 2025. Available online: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model (accessed on 12 March 2025).
23. Iqbal, J.; Kaur, R.; Stakhanova, N. PoliDOM: Mitigation of DOM-XSS by detection and prevention of unauthorized DOM tampering. In Proceedings of the 14th International Conference on Availability, Reliability and Security, Canterbury, UK, 26–29 August 2019; pp. 1–10.
24. MDN Web Docs. MutationObserver. 2025. Available online: https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver (accessed on 12 March 2025).
25. Zaini, A.; Zainal, A. Exploiting DOM Mutation for the Detection of Ad-injecting Browser Extension. In *Recent Trends in Data Science and Soft Computing, Proceedings of the 3rd International Conference of Reliable Information and Communication Technology (IRICT 2018), Kuala Lumpur, Malaysia, 23–24 June 2018*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 657–669.
26. Khodayari, S.; Pellegrino, G. It's (dom) clobbering time: Attack techniques, prevalence, and defenses. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–24 May 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1041–1058.
27. EICAR (European Institute for Computer Antivirus Research). Main Page for Anti-Malware Research. 2025. Available online: https://www.eicar.org (accessed on 12 March 2025).
28. EICAR (European Institute for Computer Antivirus Research). Anti Malware Testfile. 2025. Available online: https://www.eicar.org/download-anti-malware-testfile/ (accessed on 12 March 2025).
29. AMTSO. Anti-Malware Testing Standards Organization, Main Page for Anti-Malware and Phishing Test Tools. 2025. Available online: https://www.amtso.org/ (accessed on 12 March 2025).
30. AMTSO. Anti-Malware Testing Standards Organization, Phishing Test Page. 2025. Available online: https://www.amtso.org/feature-settings-check-phishing-page/ (accessed on 12 March 2025).
31. AV-TEST. The Independent IT-Security Institute. 2025. Available online: https://www.av-test.org (accessed on 12 March 2025).
32. VirusTotal. Free Online Virus, Malware and URL Scanner. 2025. Available online: https://www.virustotal.com (accessed on 12 March 2025).
33. Laperdrix, P.; Bielova, N.; Baudry, B.; Avoine, G. Browser fingerprinting: A survey. *Acm Trans. Web TWEB* **2020**, *14*, 1–33. [CrossRef]
34. Englehardt, S.; Narayanan, A. Online tracking: A 1-million-site measurement and analysis. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1388–1401.
35. Karami, S.; Ilia, P.; Solomos, K.; Polakis, J. Carnus: Exploring the Privacy Threats of Browser Extension Fingerprinting. In Proceedings of the 27th Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 23–26 February 2020.
36. Zhang, D.; Zhang, J.; Bu, Y.; Chen, B.; Sun, C.; Wang, T. A survey of browser fingerprint research and application. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 3363335. [CrossRef]

37. Nikiforakis, N.; Kapravelos, A.; Joosen, W.; Kruegel, C.; Piessens, F.; Vigna, G. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 19–22 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 541–555.
38. Statcounter GlobalStats. Desktop Browser Market Share Worldwide. 2025. Available online: https://gs.statcounter.com/browser-market-share/desktop/worldwide (accessed on 12 March 2025).