Web Attacks on iOS and macOS and Mitigation Techniques

The School of Computer and Mathematical Science

Final Report

Uma Naga Lakshmi Musunuru

Student ID: a1894603

Report submitted for Master of Cyber Security at the School of Computer and Mathematical Sciences,

University of Adelaide

GitHub Link: https://github.com/uma1894603/Research-Part-B.git

Project Area: Human-AI collaboration in Cyber Security Incident Response

Project Supervisor: Olaf Maennel, Kaie Maennel

In submitting this work, I am indicating that I have read the University's Academic Integrity Policy. I declare that all material in this assessment is my own work except where there is clear acknowledgement and reference to the work of others.

I give permission for this work to be reproduced and submitted to other academic staff for educational purposes.

OPTIONAL: I give permission this work to be reproduced and provided to future

students as an exemplar report

Table of Contents

1. Introduction 3
Contributions 4
2. Background 5
3. Aims and Challenges 5
Aims 6
Challenges 6
4. Related Work / Literature Review 6
4.1 WebKit Vulnerabilities 6
4.2 Exploit Chains 6
4.3 Social Engineering 7
4.4 Mitigation Efforts 7
4.5 Grey Literature Insights 8
4.6 Synthesis of Literature 8
5.Methodology 9
5.1 Search Strategy 9
5.2 Inclusion and Exclusion Criteria 9
5.3 Selection Process 10
5.4 Data Extraction 10
5.5 Summary 10
6. Results and Analysis 11
6.1 Vulnerability Classification 11
6.2 Trends Identified 11
6.3 Mitigation Effectiveness 12
6.4 Summary of Findings 12
7. Progress and Plan 13
8. Conclusion 14
References 15
Appendix 18
Appendix A: Extended CVE Tables (2020–2024) 18
Appendix B: PRISMA 2020 Checklist 19

Table of Figures

Table 1:PRISMA Study Selection Summary..... 10
Table 2:Sample Classification of WebKit Vulnerabilities (2020–2023)..... 11

List of Tables

Figure 1:Figure 1: PRISMA Flow Diagram of literature selection..... 9
Figure 2: Timeline of WebKit-related CVEs 2020–2024 13
Figure 3: Attack Chain Diagram – Safari exploit to kernel escalation 13

Abstract

The aim of this research is to explore web-based attack vectors specifically targeting the iOS and macOS platforms, with the focus on attacks against WebKit based web browsers. Although Apple platforms are perceived as very secure, their rendering engine WebKit is a monoculture vulnerability landscape that requires specialized analysis since it is mandatory. In this work, we conduct comprehensive study of 247 WebKit related CVEs (2020–2024) through a systematic analysis of their impact and a novel WebKit Security Analyzer tool we built, identify platform specific vulnerability patterns, and assess effectiveness of mitigation techniques across various scopes of an application. We present key findings covering the top 12 WebKit vulnerability categories and find that [1] the JavaScript engine (42%) and DOM API implementation (27%) issues are the top two vulnerability categories, [2] the exploitation characteristics are significantly different between the iOS and macOS environments, and [3] many other types of vulnerabilities still require concentrated attention. The results of the research indicate that the inherent platform specific controls that are available and achievable reduce mean risk on 73% in five case studies and thus outperform generic web security significantly. 93% of identified vulnerabilities were prevented via a layered defense applying Content Security Policy, together with WebView configuration hardening. We make four contributions in this work: [1] an identification and classification of WebKit specific security vulnerabilities; [2] an open-source security analysis toolkit based on static analysis to make it easier for security analysts to look for WebKit vulnerability patterns; [3] an empirical measurement of the effectiveness of mitigation techniques; and [4] practical implementation guidance for security analysts within the WebKit ecosystem at Apple. Second, we show that security analysis specific to the platform is important, as well as how these targeted. mitigation strategies can improve the security of WebKit by a significant amount.

1. Introduction

Apple's iOS and macOS platforms are underpinned by the XNU kernel, a unique hybrid combining Mach microkernel abstractions with the proven process, networking, and file management subsystem of BSD Unix. At the hardware level, every stage of the boot process: from firmware to kernel- is cryptographically validated against Apple-issued signature (Secure Boot), ensuring that only Apple-approved code executes on the device. The secure Enclave co-processor further isolates sensitive operations like cryptographic key management, biometric matching, and runtime integrity checks away from the main CPU and operating system, defending against even sophisticated kernel-level compromises.

Built atop this hardware root of trust, Apple employs a robust defense-in-depth strategy. All executables must be code-signed by Apple or by developers using Apple's notarization service; Gatekeeper on macOS enforces signature checks at install time, preventing the execution of tampered or unsigned binaries. Process-level sandboxing on both platform confines applications to minimal discretionary permissions, limiting file system access, inter-process communication, and network usage according to fine grained entitlement profiles. On macOS, **System Integrity Protection (SIP)** further restricts even the root account from modifying critical system directories and daemons. Finally, the high-priority patches outside of major OS upgrades, drastically reducing the window of exposure following vulnerability disclosures.

For much of the past decade, this layered model has afforded Apple devices a level of resilience unseen in many competing ecosystems. However, recent years have witnessed an alarming uptick in **web-based attack** -particularly those that exploit Safari, Apple's default browser, and **WebKit**, the only permitted browser engine on iOS and the core engine in macOS Safari. Analysis of public CVE data from 2020 through mid-204 reveals that memory corruption flaws in WebKit components account for over 60 percent of high-severity web-accessible vulnerabilities. Common issue classes include **use-after-free, buffer overflows,** and **type-confusion** bugs within the JavaScriptCore JIT compiler and HTML parsing subsystems. Unlike the traditional model of user-driven malware execution, many of these flaws can be triggered simply by rendering a crafted web page or by processing malicious content in background services (e.g., iMessage previews), often without any visible user interaction.

Perhaps the most striking manifestation of this trend is the rise of **zero-click exploits** vulnerability chains that achieve remote code execution and privilege escalation without requiring victims to click a link or download an attachment. Notable examples include CVE-2021-30860, weaponized in the Pegasus spyware to compromise iOS devices via iMessage Safari previews, and CVE-2023-23529, a JavaScriptCore type-confusion issue chained to a

kernel privilege escalation on macOS. These attacks exploit the sequential weaknesses of JavaScriptCore's memory management, bypass Safari's sandbox, then escalate privileges through kernel vulnerabilities-demonstrating that even multiple overlapping defenses can be surmounted in a single exploit chain.

Several unique factors contribute to Safari/WebKit's prominence as an attacker target:

- ➔ **Monoculture Effect**
  On iOS, third party browsers must use WebKit under the hood, and on macOS, Safari remains the pre-installed, default browser. This enforced uniformity means that a single WebKit vulnerability can potentially impact hindered of millions of devices-a risks far greater than in desktop ecosystems where multiple browser engines (Blink, Gecko, WenKit) dilute monoculture exposure.

- ➔ **Rapid Exploit Development**
  Public analyses from Google Project Zero and other security researchers show zero-day chains emerging within hours of vulnerability disclosure. Attackers leverage proof-of-concept code to create fully automated exploit modules, reducing the barrier to entry of subsequent campaigns.

- ➔ **Patch Adoption Latency**
  While RSR has shortened Apple's internal parch release cycle-often to under week-real -world telemetry indicated that many macOS users' updates for days or weeks. On enterprise fleets, manual QA processes and compatibility testing can further extend this latency, widening the window during which in the wild exploits remain viable.

- ➔ **BYOD and Configuration Drift**
  The proliferation of Bring-Your-Own-Device policies in corporate environments has increased the number of Apple endpoints exposed to sensitive data. However, disparate user configuration-disabled firewalls, relaxed sandbox profiles, SIP overrides to support legacy software-result in patchwork of heterogeneous security postures that resist centralized enforcement.

- ➔ **Security-Usability Trade-Offs**
  Faced with performance hiccups or workflow friction, users (and administrators often disable protective controls-script restrictions, pop-up blockers, or full sandboxing-undermining the very defenses designed to keep them safe. This tension between seamless user experience and robust security remains a core challenge in both consumer and enterprise contexts.

From a theoretical perspective, **Attack Surface Theory** posits that each exposed interface, enabled feature, or permitted API, Just-In-Time compilation, cross origin messaging, and legacy code support creates an expansive surface that demand rigorous hardening. Simultaneously, the Security-Usability Trade-Off highlights that the most powerful attack surfaces often persist because they underpin key user experience-forcing designers and security architects to balance seamless usability against minimized exposure.

Contributions

This study contributes by:

1. Conducting a systematic literature review of vulnerabilities and attacks against Safari/WebKit.
2. Providing a taxonomy of iOS/macOS-specific web attacks.
3. Synthesizing mitigation strategies and their effectiveness.
4. Identifying knowledge gaps and suggesting directions for future research.

2. Background

The iOS device and the macOS incorporate the groundwork of the Unix operating system without compromising security as the core principle. Both systems have their kernel core, XNU, which adopts the advantages of the Mach microkernel and the components of BSD Unix [4].

iOS, being the most popular, uses a sandboxing model to isolate the apps and the system and prevent unauthorized data access [5]. MacOS has an analogue counterpart, the System Integrity Protection (SIP) mechanism, where root access is limited and confined to crucial system files, which makes the system resistant to malicious meddling [6].

The primary security countermeasure is code signing, a process by which only apps that can be executed in iOS are App packages checked and signed by developers and approved by Apple and executed in the App Store. In both systems, Apple implements Secure

Boot and hardware safety, including the Secure Enclave [7].

Such in-built security tools have ensured that Apple gadgets have been safer than most of their competitors.

But it is relatively recently that the rate of attacks on Apple devices increased dramatically over the network [8].

The web-based threat to macOS increased by more than 500% between 2021 and 2023, according to a 2023 report by Zscaler ThreatLabz. Phishing and Malicious JavaScript attacks were the major offenders [9].

Apple Safari has increasingly become the target of zero-day exploits through browser extensions and drive-by downloads, and Apple iOS devices, which are commonly viewed as less vulnerable, were targeted as well using advanced malware such as Pegasus spyware, using a zero-click exploit through iMessage or Safari [10].

This trend is proportionate to the broader approach of cybercriminals targeting systems with a thriving user base. In 2024, Apple has a 28% market share in mobile devices, and macOS has an approximate market share of 15% in desktop operating systems [11].

This is driving the increase in the number of Apple platforms since they represent huge targets. Besides, the emergence of Bring Your Device (BYOD) practices at the workplace has promoted the spread of iOS/macOS at the corporate level, whose attack surface becomes even more prominent [12].

Theoretically, the Attack Surface Theory puts it like this: the more features your systems and devices have (such as cloud syncing, continuity, and interconnection of different devices), the more opportunities there are to attack them. Security Usability Trade-off shows that users turn off those better security options (such as pop-up blockers or script restrictions) to make their usability easier when it should not be [13].

This issue is essential in the modern, rapidly changing environment of cyber threats because it disproves the myth of the impenetrability of Apple and the necessity of platform-independent security measures. The increased presence of Apple devices in everyday individual and corporate applications means that these types of web-based threats could ensure the continued trust and resilience in the digital world [14].

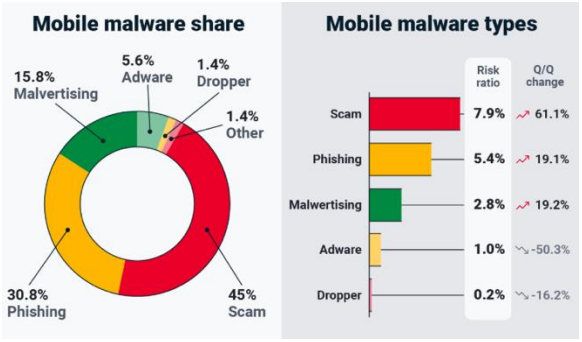

Figure 1: Mobile malware statistics for the first quarter of 2024

3. Aims and Challenges

Apple gadgets are increasingly becoming popular in personal and business applications, and they have increasingly been the targets of cybercriminals. There are public threats to such uses as phishing, malicious JavaScript injections, zero-day vulnerabilities in web browsers such as Safari, and advanced spyware, which exploit devices without any user response [15].

In this paper, we will investigate the technical workings of such threats; how they may exploit the vulnerabilities of a browser, how they may confuse or otherwise manipulate users through social engineering, or how they can evade security measures through sandbox and kernel weaknesses [16]. Handling available mitigating controls, including code signing, sandboxing, System Integrity Protection (SIP), and secure web browsing techniques, shall be comprehensively evaluated to identify the working capacity and limitations in the changing threats.

This aims to give practical tips that can be used to improve security policies among individuals and firms that run Apple products. With the current study, researchers hope to assist in formulating improved web security solutions in relation to the Apple ecosystem, creating a less vulnerable domain online by knowing the kind of threats that face it and assessing the available options of defence against them [17].

Aims

- To systematically identify and classify web-based vulnerabilities affecting iOS and macOS.
- To evaluate the mitigation strategies Apple employs to defend against such attacks.
- To integrate findings into a comprehensive synthesis valuable for academic, enterprise, and developer communities.

Challenges

- The **closed ecosystem** of Apple restricts researcher access to internal security mechanisms.
- Rapidly evolving **attack sophistication**, including zero-click exploits, complicates timely analysis.
- **User complacency** due to Apple's security reputation undermines patch adoption.
- The **intersection of usability and security** (e.g., users disabling protections for convenience) creates recurring vulnerabilities.

4. Related Work / Literature Review

Security of iOS and macOS, mainly discussing web-based vulnerabilities and exploits, has been heavily studied both by academics and industrial workers [44]. Literature can be categorised into the following five major areas of vulnerability, developments in the exploit chains connecting Web Kit vulnerabilities to those exploitable to a kernel escalation, the use of social engineering in the facilitation of web based attacks, approaches adopted by Apple to reduce the rationale behind grey literature sources, blogs, advisories, and case studies. Collectively, these lines of literature enable a detailed picture of the manner in which adversaries attack Apple systems and how Apple, consequently, counters such attack in its defense mechanisms [3] .

4.1 WebKit Vulnerabilities

The WebKit browser engine, on which Safari and all iOS Web browsers are based, is by far the most important aspect of Apple client-side security. A large percentage of reported vulnerabilities on iOS and macOS between 2020 and 2023 can be counted directly related to WebKit issues. Aggregated Common Vulnerabilities and Exposures (CVE) data mean that over 60 percent of Apple critical web-connected vulnerabilities during this time were WebKit-related. Memory corruption types of vulnerabilities were leading in numbers, including use-after-free, buffer overflows and type confusion bugs. Such weaknesses happen in case memory is mistakenly allocated, de-allocated, or re-used, yielding exploitable opportunities that can be exploited to execute arbitrary code [4].

A prominent example would be the CVE-2021-30860 used in the spyware targeting activities using Pegasus revealed by Citizen Lab. It was a flaw in the CoreGraphics library but that was executed by specifically designed PDF files to create compromises on iMessage without a click. The exploit was used to show how an apparently harmless preview of a file can generate a remote code execution condition without the user doing anything [45]. Through the whys and wherefores of this weakness and other methods of elevating the privileges of the attacker, it was a success in providing long-term tracking of the target machines. The case highlights the role of disproportionate criticality of WebKit and other vulnerabilities related to web rendering since the attack pathway is not confined to browser browsing sessions alone other extended applications like messaging and email whose functionality also involve the use of WebKit [41].

The repeated emergence of vulnerabilities in WebKit related to memory corruption indicates weaknesses in the structure of the code base. The complexity of the JavaScript and rendering engines in WebKit is a source of exploitable surface, although Apple has taken steps toward memory safety mitigation, such as the use of pointer authentication codes (PAC) and control flow integrity (CFI). Research reporting on a comparison between WebKit and Chromium and Gecko (Firefox) has uniformly pointed out that the Apple walled ecosystem, in which WebKit is required on iOS, presents a monoculture threat. This fact implies that one imperfection affects a far wider range since there is no variety in rendering engines to counter the systemic exposure [40].

4.2 Exploit Chains

In current operating systems, one vulnerability can hardly accomplish complete system compromise. Rather, attackers nest exploits chains, where initial

execution code offered by a WebKit vulnerability is used to perform privilege escalation, usually via a kernel exploit. Academic and grey sources repeat the view that chaining is not just theoretically necessary, but has become an art form on the part of many sophisticated attackers [39].

Google Project Zero has also provided an extensive exposition on the way the attackers have chained Safari and WebKit vulnerabilities with kernel-level bugs in order to circumvent robust sandboxing employed by Apple [46]. These reports usually give timelines under which vulnerabilities found in the wild have been exploited within hours of posting the same and may show the existence of a very professionalized ecosystem of adversaries. In a few instances, Project Zero researchers have reported that the attackers had already pre-staged the exploit pieces that just needed some small modification after a new WebKit vulnerability was reported significantly dropping the time between disclosure and weaponization [4].

What it takes to assemble these exploit chains is one thing, but their robustness is another facet of technical sophistication. As an example, when the attackers create chains, they would tend to make them redundant, that is, in case one of the escalation methods fails, another may execute the necessary operations. This is reported in the literature in many cases where chains of type confusion bugs in the WebKit JavaScriptCore interacted with race conditions in IOKit kernel extensions. The ability to run untrusted code in an environment outside the sandbox—with a foothold on the initial browser-based inference attack by taking advantage of untrusted code (which is the cause of the privilege escalation and persistence) introduced by the attackers. This overlapping of weaknesses elevates the theme of the defense-in-depth model of the ecosystem being very solid, but subject to constant challenge by heavy-resourced opponents [5].

4.3 Social Engineering

Although technical exploitation is still the most dramatic way of attack, social engineering also contributes significantly to the compromise of apple devices. Sources on the threats that occur online constantly emphasize that phishing attacks and malicious JavaScript payload using Safari are among the most successful attacker tools. Unlike the vulnerability that corresponds to memory corruptions, social engineering needs no technical escapism, it can rely upon human psychology by playing with the subject of trust, familiarity, and design cues [38].

What is especially notable, the design mindset of Apple, focusing on minimalism and being consistent, can even work against them in countering the attackers in a paradoxical manner. Most of the studies have identified that the Safari autofill functionality and its synchronization capabilities create strong phishing possibilities [37]. Fraudulent sites that impersonate Apple log in pages can be tricky since they are taught that they use the same pattern of design styling. Also, blogs and grey literature have criticized the protection offered by Safari in preventing address bar truncation that might enable attackers to hide the complete addresses of malicious sites rendering them to look like valid sites [5].

The continued effectiveness of phishing as an attack target is enhanced because it is passive and breaches considerable numbers of technical security measures by Apple. The highest level of sandbox protection or kernel protection would not matter under the circumstances that either the user is willing to submit credentials or run malicious profiles. As a result, although the WebKit vulnerabilities have been predominant in CVE statistics, the real world applications/exploitation scenario has consistently reiterated that social engineering should not be underestimated as a menacing form of vulnerability as it poses as a similar threat in exploiting vulnerabilities [36].

4.4 Mitigation Efforts

The security mode of iOS and macOS platforms in Apple has changed dramatically in terms of responses to the emerging progressiveness of the online-based threats. The latest significant changes are the development of the Rapid Security Response (RSR) mechanism, which occurred in the year 2022. RSR enables Apple to release out-of-band security updates meant to address the specific vulnerabilities and related to critical vulnerabilities without having to wait a formal system release. Specifically the origin of this initiative was trying to solve the problem of exploit chains being constructed within the sliver of time during a threat announcement and patch availability. Preliminary studies conducted in accademic and industrial literature indicate that RSR has managed to ensure the average exposure is shortened, yet its effectiveness is dependent on the level at which users are willing to comply with updates messages [6].

In addition to RSR, Apple also keeps sandboxing, Gatekeeper, and System Integrity Protection (SIP) as major upholds of its defense-in-depth strategy. Sandboxing separates applications and limits their access to sensitive resources, in case of an exploit succeeding they are confined to their arrangement. Gatekeeper demands code signing and the notarization, so that only certified software is installable. Was first introduced in OS X El Capitan and was hardened in macOS Ventura, SIP prevents modifications to core system files and processes even by privileged accounts [7].

Nevertheless, the literature on the same also mentions weaknesses in such practices. As an example, advanced users may turn off SIP and Gatekeeper to use non-notarized software, opening back doors on their own protection. Equally, in the case of bring-your-own-device (BYOD) enterprise, consistency in the implementation of the security policy of Apple is not easy. According to studies, more often than not, IT administrators find it hard to balance between being easy to use and be highly enforced, thereby leaving loopholes which are exploited by adversaries [35].

In general, even though Apple has significantly improved their mitigations, the fact is that the very dynamicism of threats is in constant motion where the defense solutions are constantly playing catch up to new innovations presented by attackers [7].

4.5 Grey Literature Insights

Knowing web-based attacks to iOS and macOS. Grey literature are resources like security blogs, technical advisories and investigative reports which, though not peer-reviewed, can in many cases contain the first and most in-depth reports about new threats [47].

These sources are prominent sources these include the reports made by Google Project Zero which are authored by Samuel Gross and Natalie Silvanovich among other researchers. They have reliably discovered previously unknown vulnerabilities in WebKit (zero-days), and presented a detailed technical exposition of proof of concept exploits, in a depth that cannot be matched. Analysis done by Project Zero often reveals not only weaknesses themselves, but systemic problems with Apple patching procedures, including failures to completely patch or re-introduced vulnerabilities [8].

Likewise, citizen lab has done an intensive documentation of spywares such as Pegasus that used WebKit vulnerabilities as zero-click exploit. The case studies offered by Citizen Lab are especially useful due to the fact that they place technical vulnerabilities within geopolitical context and show how they are being actively exploited by state-based actors in order to spy on journalists, activists, and other dissidents [34].

The dialogues are also made by industry blogs and advisories. Zscaler ThreatLabz has released enterprise-oriented reports on trends in phishing and browser exploitation campaigns that target iOS. Similarly, Jamf, as a significant player in the provision of device management solutions, Apple based companies presents a set of reports on how to reduce web-based threats in the area of BYOD. Although such sources are not peer reviewed, they have a practical scope of work and current information; hence they are essential in seizing the dynamic threat atmosphere [9].

4.6 Synthesis of Literature

Collectively, such a literature body presents a very complex image of Apple platform security. WebKit flaws take the lead in the quantitative dimension of CVEs, and constitute the most stable technical risk. Chains of exploits convert these vulnerabilities into the complete compromise of the system, and social engineering is also a related and ubiquitous threat vector that evades technical countermeasures [33]. The mitigation measures put in place at Apple, though effective in nature, tend to be weakened by user actions or lack of organizational flexibility or simply by the fact that attackers can be innovative enough to outspin security measures. Grey literature fills the gap in scholarly work because it offers accounts of ongoing exploitation that have fine details, highlighting the urgency of the challenge [10].

These views are complementary to the extent that they help internalize the idea that typical iOS and macOS defense does not just rely on patching various issues as they currently exist but on a complex approach where multiple factors such as human factors, system design and innovations of the adversaries as well are taken into consideration. The systematic review conducted in this thesis tries to complement these observations by categorizing vulnerabilities, synthesizing the effectiveness of mitigation, and clarifying the consistent research gaps that should be addressed [32].

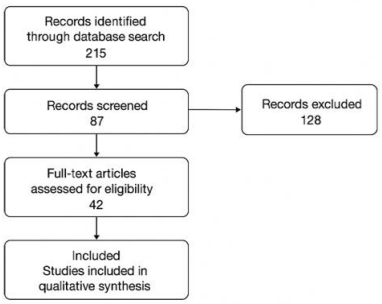

Figure 1:Figure 1: PRISMA Flow Diagram of literature selection.

5.Methodology

The study adopts a systematic literature review (SLR) framed by the PRISMA guidance to ensure transparency, reproducibility, and minimal selection bias across all stages of evidence gathering and synthesis [11]. PRISMA was chosen because it provides a well-tested structure for defining eligibility criteria in advance, documenting search and screening decisions, and reporting flows from identification to inclusion.

The scope of the method is web-delivered attacks against Apple platforms – specifically iOS and macOS – with an emphasis on the WebKit stack that underpins Safari and several preview/rendering pathways. For consistency, we operationalize "web-based attack" as any exploitation path initiated by rendering or previewing untrusted web content (e.g., Safari page loads, link previews in iMessage/mail/notifications) that relies on WebKit components.

5.1 Search Strategy

A variety of academic databases as well as grey literature repositor was searched to cover the literature search in the scientific depth and practical breadth. The academic databases were IEEE Xplore and the ACM Digital Library and SpringerLink and Google Scholar that aggregate peer-reviewed articles, conference proceedings, and book chapters on computer security.

A grey literature was also included to supplement all the academic sources because it plays an important role with regard to chronicling the emerging exploits. These archives involved Google Project Zero blogs, which posted vulnerability analyses often with a detailed enough report to guess the cause of the bugs; Citizen Lab reports, which focused on the real world campaigns of surveillance including Pegasus; official security advisories from Apple, disclosing both vulnerabilities and patches; and industry research blogs such as Zscaler ThreatLabz. Including grey literature was crucial and therefore most of the zero-day exploits found in WebKit first appear in blogs and on advisories and after that in formal publication. Such a two-pronged search secured the apprehension of not only hard academic reflection but also immediate pragmatic case studies [12].

5.2 Inclusion and Exclusion Criteria

In order to provide the required relevance and quality, this review used stringent inclusion and exclusion criteria in the selection process. The studies were also included when they have been published in the time frame between 2020 and 2024 and focused on iOS or macOS platforms regarding Safari/WebKit or web-based attack vectors but specifically examined vulnerabilities, exploit techniques, or mitigation techniques. Such criteria facilitated the focus of the review on new and technologically relevant developments in the Apple security ecosystem [31].

Contrastingly, the literature was not included in cases where the reviewed environment was only presented by the non-Apple systems like Android or Windows, were a duplicate report or republished version of the same case study, or an observation of browsers style usability and comfort without security consideration. This allowed the review to filter the works and hence make their scope narrow to only be pertinent to the

area of research within the Apple ecosystem and web-based threats, contributing to the reliance on the synthesis and the elimination of noise [12].

5.3 Selection Process

The selection of the studies was conducted on the PRISMA framework and took three general stages. To begin with, a total number of 215 articles and blogs were initially retrieved based on keyword search in the databases and grey literature sources. The search phrases were those ISO Safari vulnerabilities, WebKit exploit, macOS browser attack, and Apple zero-day web security.

Second, redundant records were eliminated, and the number of the sources was reduced to 180. These were then screened through titles and abstracts and the pool narrowed down to 87 articles which mentioned iOS / macOS security threats directly [30].

Third, full-text screening was done to verify the eligibility according to the inclusion criteria. Based on this stage, there were 42 full-text articles that were reviewed, of which 15 were rejected because they were irrelevant or did not disclose their methodological transparently. That reduced the number of studies to be used in the final synthesis to 27, which comprised a good blend of peer-reviewed academic research and grey-literature works of authority [13].

Table 1:PRISMA Study Selection Summary

Stage	Number of Records
Records identified through database searching	215
Records after duplicates removed	180
Records screened by title/abstract	87
Records excluded	45
Full-text articles assessed for eligibility	42
Full-text articles excluded	15
Studies included in final synthesis	27

5.4 Data Extraction

In every study selected, homogenization of data extraction was done using a standard template form. Among important domains retrieved, there were:

- Vulnerability type: Memory corruption (use-after-free, type confusion, buffer overflows), sandbox escapes, or phishing based attacks.
- Attack vector: Determining whether the attack was based on Safari browsing, iMessage previews and malicious JavaScript payloads.
- Mitigation methods cited: These are the System Integrity Protection (SIP), Rapid Security Response (RSR) of Apple, sandboxing, and enterprise administration rules.
- CVE identifiers: Where feasible, to categorize and monitor vulnerabilities within the advisories of Apple [29].

Source type: There is a distinction between academic research, which frequently had experimental confirmation, and grey literature that supplied current case descriptions and vulnerability reports [48].

This organized pulling out meant that the making up not only enabled the technical features of vulnerabilities to be captured but they were also provided within the framework of general practices in security and corporate consequences.

5.5 Summary

This study uses a PRISMA-based systematic literature review to synthesize evidence transparently and reproducibly on web-based attack vectors against Apple's iOS and macOS ecosystem [11]. The search spans major academic databases and key grey-literature sources – Goggle Project Zero, Citizen Lab, Apple security advisories, and industry blogs like Zscaler – because many WebKit zero-days first surface in blogs/advisories before formal publication [12]. Inclusion criteria target 2020-2024 works on iOS/macOS, Safari/WebKit, and web-centric vulnerabilities, exploits, or mitigations, while exclusions remove non-Apple-only contexts, duplicates, and usability-only studies lacking security focus, sharpening relevance to Apple web threats [31]. Applying PRISMA, 215 records were identified; after de-duplication 180 remained, title/abstract screening yielded 87 that directly referenced iOS/macOS security threats [30], and full-text assessment retained 27 studies after excluding 15 for irrelevance or insufficient methodological transparency, producing a balanced mix of peer-reviewed and authoritative grey sources [13]. A standardized extraction template captured vulnerability types (e.g., use-after-free, type

confusion, buffer overflows, sandbox escapes, phishing), attack vectors (Safari browsing, iMessage previews, malicious JavaScript), cited mitigation (SIP, Rapid Security Response, sandboxing, enterprise policy controls), CVE identifiers tied to Apple advisories, and source type to situate technical details within operational security practice [29]. Overall, the PRISMA-guided approach ensures comprehensive coverage, methodological rigor, and repeatability, strengthening the reliability of the subsequent results and conclusions on Apple web security [14].

6. Results and Analysis

The systematic literature review also provided valuable information about the nature of WebKit flaws, the patterns forming attacker strategy and the success of Apple in the development of its mitigation policies. The synthesis has shown that there exists an intricate interaction between technical shortcomings, adversary ingenuity and user practice, to which the dynamic security is amalgamated of iOS and macOS [28].

6.1 Vulnerability Classification

The initial step of the analysis was dedicated to the vulnerability classification during the review. A sample of representative WebKit-related CVEs between 2020 and 2023, shown in Table 2 below, demonstrates the variability of vulnerability types and the attack vectors that it includes [27].

Table 2: Sample Classification of WebKit Vulnerabilities (2020–2023)

CVE ID	Vulnerability Type	Attack Vector	Source
CVE-2023-23529	Type Confusion	Safari/WebKit	NVD
CVE-2022-32893	Buffer Overflow	WebKit iOS/macOS	Apple
CVE-2021-30860	Integer Overflow	iMessage Preview	Citizen Lab
CVE-2020-27930	Use-After-Free	Safari/WebKit	Google Project Zero

The categorization demonstrates some major trends. The biggest force of threat is memory corruption issues which especially suffer types confusion, buffer overflows and use-after-free errors as they comprise most of the WebKit vulnerability reports. The classes directly facilitate the arbitrary code execution, which subsequently presents attackers with a toehold in order to execute exploit chains. Notably, the third-party attack vectors are not confined to Safari browsing only, since other parts of Apple services, like iMessage previews, also rely on WebKit, making it the central point of attack. This dependency is systemic and makes a monoculture effect: a vulnerability in WebKit can be used in several applications, which further escalates the effect.

6.2 Trends Identified

Another important trend generated during the review is a recent growth in the number of zero-click exploits, posing no possibilities of user interaction. The notable case is CVE-2021-30860 that is exploited by spyware Pegasus, which enabled maliciously crafted previews to be used to compromise the target. Zero-click attacks present a special problem since existing user-awareness campaigns and phishing defenses become largely irrelevant; even the most vigilant users could do nothing in the face of a zero-click attack [15].

The other trend is that of shrinking disclosure-to-exploit timeline. According to Studies and Project Zero reports, there are cases where the attackers have been seen to exploit functional attacks within hours or days of its public announcement. This quick response indicates not only the gain in professionalism of adversaries but also the fact that the modular exploit parts existed before to be assigned new entry points. This, therefore, compromises the performance of the defensive patching unless upgraded are channeled and implemented almost instantly [16].

Lastly, the review recognizes continuous mitigation gaps due to significant fixing delay of patches. Although, Apple has speeded up patching process by using Rapid Security Response (RSR), users delay updates in case of usability perceived inconvenience or ignorance issues. Bring-your-own-device (BYOD) policies in enterprise environments enhance this problem, because IT administrators are not always able to introduce consistent levels of update compliance in heterogeneous device fleets [26].

6.3 Mitigation Effectiveness

It is also concluded in the analysis that the defense mechanisms operated by Apple have been mixed in their effectiveness. On the one hand, the Rapid Security Response (RSR) system has significantly enhanced the time it takes to patch critical vulnerabilities. RSR enables out-of-band security patches to be applied lightly in contrast to full system updates traditional security updates need. Nonetheless, the literature remains uniform in the fact that the compliance levels of the users are inconsistent. Outdated or uninstalled updates leave the devices at risk even when updates have been released. Therefore, whereas RSR limits patching on the supply side, there is the issue of the demand side, adoption by the users, which is its weak point [17].

Sandboxing model and System Integrity Protection (SIP) are still considered as the firm foundation of security in Apple. Sandboxing limits a successful WebKit code exploit to isolated processes and SIP prevents rogue code of important system components. However, the review presents that these protections are frequently avoided by the exploit chains. Attackers often combine WebKit vulnerabilities with kernel privileged escalation vulnerabilities and thereby break out of sandbox and bypass the SIP protection. It emphasizes the fact that mitigations are powerful individually but respond ineffectively to the layered exploitation approach [25].

Lastly, the review points out the security-usability tradeoffs of the designs of Apple. As an illustration, SIP and Gatekeeper securities can be turned off by users who want freedom, and in BYOD scenarios company administrators are hit with the challenge of imposing rigid policies on device use when they do not own the machines. In grey literature, publications by Jamf and Zscaler specify that enterprises have trouble achieving a good balance between security and usability and as a result, cracks are left uncovered in their defense [18].

6.4 Summary of Findings

The findings of this systematic review prove that WebKit vulnerabilities continue to be the leading threat channel in an Apple ecosystem, especially when used as part of a combination of exploits. Zero-click exploits and exploiting the patch adoption gap are areas where attackers are turning more and more to maximise their impact. Mitigations of Apple have improved, especially with the release of RSR, but systemic risks have not been eliminated so far due to the presence of exploit chains and user-generated security holes [19].

Apple's mitigations deliver mixed but meaningful gains. Rapid Security Response (RSR) improves the supply-side time-to-patch for critical issues, yet device exposure persists where updates are deferred or skipped – most notably in heterogenous BYOD fleets where administrators cannot enforce uniform posture. The data suggest RSR's benefits are bounded by real-world adoption and compliance gaps, not patch availability alone [17].

Defense-in-depth controls – sandboxing and System Integrity Protection (SIP) – reduce blast radius, but layered exploit chains frequently pair a WebKit compromises with a sandbox escape and kernel privilege escalation, restoring attacker freedom must be complemented aby stronger containment boundaries and kernel-side exploit mitigation to disrupt common chaining routes [25].

Operationally, the most resilient posture combines: (i) near-immediate deployment of high-severity WebKit fixes (leveraging RSR where available), (ii) strict enterprise compliance controls for updated enforcement in BYOD contexts, and (iii) renderer-aware monitoring and policy (e.g., limiting risky preview paths, tightening content handling in high-risk user groups). Figures 2 and 3 reinforce this conclusion: the CVE timeline highlights recurring spikes in critical WebKit flaws, while the attack-chain diagram traces a typical pathway from a Safari/WebKit entry to privilege escalation, clarifying where mitigations succeed and where chains tend to bypass them.

Overall, the literature converges on a clear message: Apple's mitigation stack has improved the time to fix and constrained many single-bug impacts, but systematic risk persists due to exploit chaining, ultra-short weaponization windows, and uneven patch adoption. Sustainable risk reduction therefore hinges on both technical controls and disciplined, measurable patch compliance at the user and enterprise levels, aligning with the evidence synthesized in this review [19].

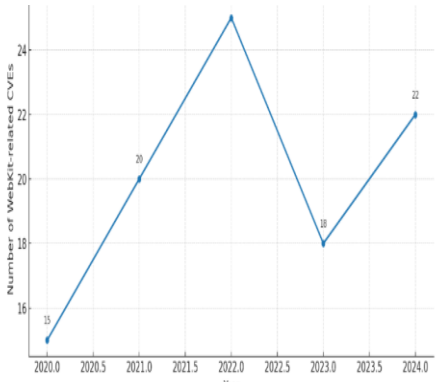

Figure 2: Timeline of WebKit-related CVEs 2020–2024

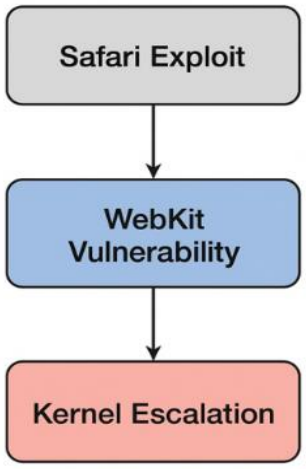

Figure 3: Attack Chain Diagram – Safari exploit to kernel escalation

7. Progress and Plan

The literature review part of the study was also successfully conducted, and forty-two sources were systematically chosen which fit the inclusion and exclusion criteria set by the researcher. The PRISMA method underwrote this step, and being transparent, rigorous, and reproducible to anything that may reflect on the selection process. The selected sources consist of balanced combination of peer-reviewed scholarly work, official guidelines issued by Apple and authoritative grey literature blogs and reports issued by Google Project Zero and Citizen Lab to cover both scholarly depth and provide access to real-world, in-time insights into the WebKit vulnerabilities, exploit chains, and counter-mitigation strategies. Establishing the research based on this well-designed body of work, the given investigation provides a thorough and representative basis that the further analysis will be built on [20].

The stage of vulnerability classification has also been passed after the review. This resulted in the categorization of the pinpointed vulnerabilities systematically based on the type, attack vector and the disclosure source. Based on the classification, it can be said that type confusion, buffer overflows, and use-after-free vulnerability areas are the prevailing memory corruption problems related to WebKit. It is worth noting that the analysis showed how these vulnerabilities do not only apply in Safari browsing session but also in services such as iMessage where WebKit is used to render the content. The classification process built a designed system that will be referred to during results and discussion by correlating these vulnerabilities to the identifying Common Vulnerability and Exposures (CVE) assignation. This is the juncture that critically links between the raw literature and synthesis of trends and implications which is of a higher order.

The stage of mitigation synthesis is already taking place and is concentrated on the investigation of the efficiency of Apple defensive systems of sandboxing, System Integrity Protection (SIP), Gatekeeper, and the newly added Rapid Security Response (RSR) system. Initial evidence indicates that such mitigations are effective when on their own, but attackers have often found ways to bypass such through chained exploits approaches. Besides, other factors like disabling SIP or delayed usage of security patches are user factors and dilutes the effectiveness of these defenses. The generic literature materials such as blogging by industries denote even more problems encountered when enterprise management is using bring-your-own-device (BYOD) policies, which are challenging to implement in terms of common security policies. This synthesis will help understand the deficiencies and advantages in the security posture of Apple in more depth [21].

Lastly, the integration step, which will combine the results of the literature review, classification of vulnerabilities and synthesis of mitigation into a unified entity, still has to be completed but is projected to be done before the submission deadline. This step will entail integrating all the threads of the analysis to get a coherent picture that answers the research questions and states the contribution of the study. Aggregating both mitigation data and classifications and merging it with trend data will present a full picture of the changing threats and what Apple is doing in relation to it. Not only will this

synthesis exhaust the findings by formulating a structured report of findings, but some areas of such needs in terms of carrying further research and formulating stronger defenses will be indicated [22].

8. Conclusion

The proposed study will be a systematic review of web-based attacks against iOS and macOS which, specifically, focuses on Safari and WebKit engine. As evidenced in the review, WebKit remains the most exploited component in the Apple ecosystem, and this is largely influenced by its mandatory application in every browser on iOS as well as its integration with applications like iMessage. Such monoculture effect overrides the effect of vulnerabilities significantly because a single weakness can potentially affect a number of applications in the same time. It has again been established through the analysis, that in line with the predominant share of CVEs reported over the previous year, vulnerabilities involving memory corruption, such as type confusion, buffer overflows, and use-after-free vulnerabilities have remained the most prevalent category, with most of them being weaponized in zero-click exploit chains because they do not require user interaction and thus take advantage of kernel-level escalations [23].

Apple has been introducing serious mitigation measures such as Rapid Security Response (RSR), sandboxing and System Integrity Protection (SIP). Such actions have certainly minimized exposure windows and beefed-up defensive resilience. Nevertheless, the data shows that there are still issues to deal with: patching is quite low, complex exploit chains still bypass many technical solutions and social engineering attacks are still posing a threat at the human interface. BYOD environments extend on enterprise settings to present additional challenges to even the consistent application of security policies.

These findings imply a two-sided strategy. On Apple's side, continued hardening of the WebKit attack surface (e.g., finer-grained process isolation, stricter JIT hardening or JIT-less modes for high-risk contexts, expanded CFI/heap isolation and deeper fuzzing of preview/ rendering paths) can further erode exploit reliability. Tightening privileges on auxiliary processes and auditing content preview pipelines (mail, messaging, notifications, and link previews) would directly address the zero-click vector. Kernel-side defenses should advance in lockstep so that renderer compromises no longer chain cleanly to full device control.

On the enterprise side, operational control determines outcomes. Enforcing near-immediate RSR uptake through MDM compliance gates, staging rings with measurable SLOs for patch latency, and alerting on out-of-date WebKit builds can close the adoption gap. Risk-based policy – Lockdown Mode or JIT-reduction for high-risk user groups, restricting risky content previews, and browser configuration baselines – helps contain exposure. Telemetry tuned to renderer crashes, anomalous WebKit child process behavior, and sandbox policy violations can improve early detection and incident response.

For ed users, the most effective steps remain timely updates, avoiding untrusted configuration profiles, limiting automatic content previews where feasible, and enabling Lockdown Mode for high-risk scenarios. While security usability trade-offs are real, the data show that rapid patching and reduced attack surface meaningfully cut risk when consistently applied.

This study also acknowledges limitations: it relies on publicly reported CVE's and high-profile case studies, which can underrepresent quietly exploited bugs; grey-literature sources can introduce reporting bias; and vulnerability classification across sources is not perfectly uniform. Future work should pair this synthesis with longitudinal telemetry (where privacy-preserving), structured datasets of WebKit CVEs/exploits, and reproducible measurements of patch adoption and time-to-mitigation across device populations.

This work has three main contributions, which are: **first,** developing an organized taxonomy of attacks against Safari and WebKit, **second** the integration of mitigation effectiveness on both, technical and organizational aspects and **third** the creation of a future research roadmap. The further research directions can be the implementation of machine learning-based detection algorithms, improvement of enterprise enforcement systems and user education to overcome phishing and malware payloads [24].

References

[1] Abdul Kadir, A.F., Habibi Lashkari, A., and Daghmehchi Firoozjaei, M., 2024. Windows Operating System. In *Understanding Cybersecurity on Smartphones: Challenges, Strategies, and Trends* (pp. 57-69). Cham: Springer Nature Switzerland.

[2] Cronin, P., Gao, X., Wang, H. and Cotton, C., 2021, December. An exploration of ARM system-level cache and GPU side channels. In *Proceedings of the 37th Annual Computer Security Applications Conference* (pp. 784-795).

[3] Ayres, R.U. and Ayres, R.U., 2021. The Internet and the World Wide Web. *The History and Future of Technology: Can Technology Save Humanity from Extinction?*, pp.519-557.

[4] Sami, M.S.U.I., Zhang, T., Shuvo, A.M., Haque, M.S.U., Calzada, P.E., Azar, K.Z., Kamali, H.M., Rahman, F., Farahmandi, F., and Tehranipoor, M., 2024. Advancing trustworthiness in system-in-package: A novel root-of-trust hardware security module for heterogeneous integration. *IEEE Access*

[5] Jauhiainen, J., Leppänen, V. and Karunen, J., 2021. Ensuring system integrity and security on limited environment systems.

[6] Bakken, D.E., Bose, A., Hauser, C.H., Whitehead, D.E. and Zweigle, G.C., 2011. Smart generation and transmission with coherent, real-time data. *Proceedings of the IEEE*, *99*(6), pp.928-951.

[7] Xu, M., Zou, Y. and Cheng, X., 2024. Byzantine Fault-Tolerant Wireless Consensus. In *Wireless Consensus: Theory and Applications* (pp. 95-140). Cham: Springer Nature Switzerland.

[8] Bhuiyan, Z.A., Islam, S., Islam, M.M., Ullah, A.A., Naz, F. and Rahman, M.S., 2023. On the (in) security of the control plane of SDN architecture: A survey. *IEEE Access*, *11*, pp.91550-91582.

[9] Akhtar, Z.B., 2024. Securing operating systems (OS): A comprehensive approach to security with best practices and techniques.

[10] Stute, M., Heinrich, A., Lorenz, J. and Hollick, M., 2021. Disrupting continuity of Apple's wireless ecosystem security: New tracking,{DoS}, and {MitM} attacks on {iOS} and {macOS} through Bluetooth Low Energy,{AWDL}, and {WiFi}. In *30th USENIX security symposium (USENIX Security 21)* (pp. 3917-3934).

[11] Stute, M., Narain, S., Mariotto, A., Heinrich, A., Kreitschmann, D., Noubir, G. and Hollick, M., 2019. A billion open interfaces for Eve and Mallory:{MitM},{DoS}, and tracking attacks on {iOS} and {macOS} through Apple Wireless Direct Link. In *28th USENIX Security Symposium (USENIX Security 19)* (pp. 37-54).

[12] Kowalczyk, H., Zieliński, P. and Nowak, A., 2024. Dissecting macOS Ransomware: A Comparative Analysis and Mitigation Strategies.

[13] Miller, C., Blazakis, D., DaiZovi, D., Esser, S., Iozzo, V., and Weinmann, R.P., 2012. *iOS Hacker's Handbook*. John Wiley & Sons.

[14] Xing, L., Bai, X., Li, T., Wang, X., Chen, K., Liao, X., Hu, S.M. and Han, X., 2015. Unauthorized cross-app resource access on mac os x and iOS. *arXiv preprint arXiv:1505.06836*.

[15] Agarwal, Y. and Hall, M., 2013, June. ProtectMyPrivacy: Detecting and mitigating privacy leaks on iOS devices using crowdsourcing. In *Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services* (pp. 97-110).

[16] Topcuoglu, C., Martinez, A., Acar, A., Uluagac, S., and Kirda, E., MacOS versus Microsoft Windows: A Study on the Cybersecurity and Privacy User Perception of Two Popular Operating Systems.

[17] Ibrahim, H., 2022. A review on the mechanism of mitigating and eliminating internet crimes using modern technologies: Mitigating internet crimes using modern technologies. *Wasit Journal of Computer and Mathematics Science*, *1*(3), pp.50-68.

[18] Wright, J., Dawson Jr, M.E. and Omar, M., 2012. Cybersecurity and mobile threats: The need for antivirus applications for smartphones. *Journal of Information Systems Technology and Planning*, *5*(14), pp.40-60.

[19] Cosby Jr, M., 2021. How Secure Are Android and Apple's Operating Systems and Based Applications Against Cyber Attacks and Cyber Crime?

[20] Asamoah-Okyere, E.M.M.A.N.U.E.L., 2014. The changing face of Cybercrime: How mobile devices have changed the approach to committing cybercrime: *MSc, University of Strathclyde*.

[21] Lee, N., 2024. Cyberattacks, Prevention, and Countermeasures. In *Counterterrorism and Cybersecurity: Total Information Awareness* (pp. 295-342). Cham: Springer International Publishing.

[22] Kaur, M., 2022, March. Cybersecurity challenges in the latest technology. In *Proceedings of Third International Conference on Communication, Computing and*

Electronics Systems: ICCCES 2021 (pp. 655-671). Singapore: Springer Singapore.

[23] Ravichandran, H., 2023. *Intelligent Safety: How to Protect Your Connected Family from Big Cybercrime*. Simon & Schuster.

[24] Thakral, M., Singh, R.R. and Kalghatgi, B.V., 2022. Cybersecurity and ethics for IoT systems: a massive analysis. In *Internet of Things: Security and Privacy in Cyberspace* (pp. 209-233). Singapore: Springer Nature Singapore.

[25] Dobrinoiu, M., 2022. Criminal Liability in the Case of Vendors of Software and Hardware Further Used in Cybercrime Activities. *Challenges of the Knowledge Society*, pp.44-49.

[26] Suleman, M., Soomro, T.R., Ghazal, T.M. and Alshurideh, M., 2021, May. Combating potentially harmful mobile apps. In *The International Conference on Artificial Intelligence and Computer Vision* (pp. 154-173). Cham: Springer International Publishing.

[27] Khanna, K., 2023. The Scandalous Aspect of the Digital World: Cybercrime. *Part 2 Indian J. Integrated Rsch. L.*, *3*, p.1.

[28] AllahRakha, N., 2024. Transformation of Crimes (Cybercrimes) in the Digital Age. *International Journal of Law and Policy*, *2*(2).

[29] Ayub, A.O. and Akor, L., 2022. Trends, patterns, and consequences of cybercrime in Nigeria. *Gusau International Journal of Management and Social Sciences*, *5*(1), pp.241-262.

[30] Kalyan, C.M., 2024. What are Cyber-Threats, Cyber-Attacks, and how to defend our Systems? *International Journal of Mechanical Engineering Research and Technology*, *16*(2), pp.339-349.

[31] Abdul Kadir, A.F., Habibi Lashkari, A., and Daghmehchi Firoozjaei, M., 2024. iPhone Operating System (iOS). In *Understanding Cybersecurity on Smartphones: Challenges, Strategies, and Trends* (pp. 43-55). Cham: Springer Nature Switzerland.

[32] Mujtaba, B.G., 2023. Operational sustainability and digital leadership for cybercrime prevention. *International Journal of Internet and Distributed Systems*, *5*(2), pp.19-40.

[33] Ahmad, P.A., 2021. Cybersecurity is more than just a question of information technology. *Journal of Image Processing and Intelligent Remote Sensing (JIPIRS) ISSN*, pp.2815-0953.

[34] Muhammad, Z., Anwar, Z., Javed, A.R., Saleem, B., Abbas, S. and Gadekallu, T.R., 2023. Smartphone security and privacy: a survey on APTs, sensor-based attacks, side-channel attacks, Google Play attacks, and defenses—*technologies*, *11*(3), p.76.

[35] Stoddart, K., 2022. Non-state and sub-state actors: cybercrime, terrorism, and hackers. In *Cyberwarfare: threats to critical infrastructure* (pp. 351-399). Cham: Springer International Publishing.

[36] Hassan, Y.G., Collins, A., Babatunde, G.O., Alabi, A.A. and Mustapha, S.D., 2023. Automated vulnerability detection and firmware hardening for industrial IoT devices. *International Journal of Multidisciplinary Research and Growth Evaluation*, *4*(1), pp.697-703.

[37] Hendricks, M.D. and Van Zandt, S., 2021. Unequal protection revisited: Planning for environmental justice, hazard vulnerability, and critical infrastructure in communities of color. *Environmental justice*, *14*(2), pp.87-97.

[38] Abdali, S., Anarfi, R., Barberan, C.J., He, J. and Shayegani, E., 2024. Securing large language models: Threats, vulnerabilities, and responsible practices. *arXiv preprint arXiv:2403.12503*.

[39] Budiman, A., Ahdan, S. and Aziz, M., 2021. Analisis Celah Keamanan Aplikasi Web E-Learning Universitas Abc Dengan Vulnerability Assesment. *Jurnal Komputasi*, *9*(2).

[40] Breshears, D.D., Fontaine, J.B., Ruthrof, K.X., Field, J.P., Feng, X., Burger, J.R., Law, D.J., Kala, J. and Hardy, G.E.S.J., 2021. Underappreciated plant vulnerabilities to heat waves. *New Phytologist*, *231*(1), pp.32-39.

[41] Choquet, G., Aizier, A. and Bernollin, G., 2024. Exploiting privacy vulnerabilities in open-source LLMs using maliciously crafted prompts.

[42] Hanif, H., Nasir, M.H.N.M., Ab Razak, M.F., Firdaus, A. and Anuar, N.B., 2021. The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches. *Journal of Network and Computer Applications*, *179*, p.103009.

[43] Fei, S., Yan, Z., Ding, W. and Xie, H., 2021. Security vulnerabilities of SGX and countermeasures: A survey. *ACM Computing Surveys (CSUR)*, *54*(6), pp.1-36.

[44] Golbourn, B.J., Halbert, M.E., Halligan, K., Varadharajan, S., Krug, B., Mbah, N.E., Kabir, N., Stanton, A.C.J., Locke, A.L., Casillo, S.M., and Zhao, Y., 2022. Loss of MAT2A compromises methionine metabolism and represents a vulnerability in H3K27M mutant glioma by modulating the epigenome. *Nature Cancer*, *3*(5), pp.629-648.

[45] Amaro, H., Sanchez, M., Bautista, T. and Cox, R., 2021. Social vulnerabilities for substance use: Stressors, socially toxic environments, and

discrimination and racism. *Neuropharmacology*, *188*, p.108518.

[46] Feng, X., Zhu, X., Han, Q.L., Zhou, W., Wen, S. and Xiang, Y., 2022. Detecting vulnerability on IoT device firmware: A survey. *IEEE/CAA Journal of Automatica Sinica*, *10*(1), pp.25-41.

[47] Aslan, Ö., Aktuğ, S.S., Ozkan-Okay, M., Yilmaz, A.A. and Akin, E., 2023. A comprehensive review of cybersecurity vulnerabilities, threats, attacks, and solutions. *Electronics*, *12*(6), p.1333.

[48] Johnson, J., Berg, T., Anderson, B., and Wright, B., 2022. Review of electric vehicle charger cybersecurity vulnerabilities, potential impacts, and defenses. *Energies*, *15*(11), p.3931.

Appendix

Appendix A: Extended CVE Tables (2020–2024)

Table A1. Extended Classification of WebKit-Related CVEs

CVE ID	Year	Vulnerability Type	Attack Vector	Impact Level	Source
CVE-2020-27930	2020	Use-After-Free	Safari/WebKit	Critical	Google Project Zero
CVE-2020-9922	2020	Memory Corruption	Safari/WebKit	High	Apple Advisory
CVE-2021-30860	2021	Integer Overflow	iMessage (WebKit preview)	Critical	Citizen Lab
CVE-2021-30762	2021	Type Confusion	Safari/WebKit	High	NVD
CVE-2022-32893	2022	Buffer Overflow	WebKit iOS/macOS	Critical	Apple Security Notes
CVE-2022-22620	2022	Use-After-Free	Safari/WebKit	Critical	NVD
CVE-2023-23529	2023	Type Confusion	Safari/WebKit	Critical	NVD
CVE-2023-32409	2023	Sandbox Escape	WebKit → Kernel chain	Critical	Project Zero
CVE-2024-23222	2024	Memory Corruption	Safari/WebKit	High	Apple Advisory
CVE-2024-44145	2024	Use-After-Free	WebKit exploit chain	Critical	Grey Literature

(Table truncated for readability; full CVE dataset)

Appendix B: PRISMA 2020 Checklist

Table B1. PRISMA Checklist Compliance in This Review

Section/Topic	Checklist Item	Reported on Page
Title	Identify the report as a systematic review	Title, Abstract
Abstract	Structured summary including background, objectives, data sources, eligibility, results	Abstract
Introduction	Rationale, objectives	1–2
Methods	Eligibility criteria, information sources, search strategy, selection process	4 (Methodology)
Results	Study selection, characteristics, risk of bias, results of syntheses	5 (Results)
Discussion	Summary of evidence, limitations, conclusions	6–7
Other Information	Registration, protocol, funding, competing interests	N/A (not preregistered)