Vulnerabilities, Exploits, and Countermeasures in Sandbox/Closed System Environments: A Focused Review on AI and Generative AI

UO Maqubela [220098304]

[1] University of Johannesburg Auckland Park Kingsway Campus 2000

Abstract. This review delves into the weaknesses and defences, in controlled environments like sandboxes and closed systems with an emphasis on AI and generative AI fields. It sheds light on the vulnerabilities linked to machines (VMs) hardware components as well as software, in sandboxed setups and the cyber threats aiming at these sectors. Furthermore, it examines approaches to minimize risks and protect against dangers based on research findings and real-world examples.

Keywords: Sandboxed Environments, Software Vulnerabilities, Large Language Models, Hardware Vulnerabilities

1 Introduction

In the realm of computing technology lies a practice known as sandboxing that plays a role, in keeping applications to shield the main system and other operations from potential harmful impacts of code execution meddling with their functioning. This approach is particularly significant in intelligence (AI). Generative AI domains where the adoption of sandbox environments helps confine AI behaviors within a controlled space to mitigate any potentially detrimental outcomes. In this examination piece the primary goal is to pinpoint and elucidate weaknesses, vulnerabilities and defensive strategies relevant, in safeguarding environments within the AI landscape.

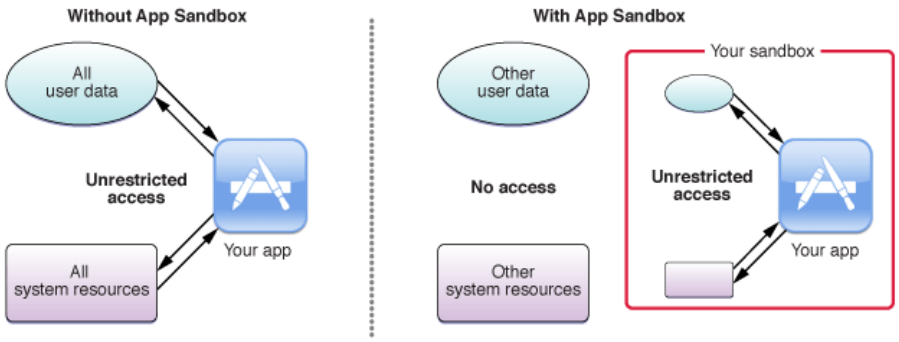

Fig. 1. This image shows how Sandboxing occurs, describing precisely how applications are restricted from the rest of the system ultimately protecting user data and system vulnerabilities.

2 Virtual Machine Vulnerabilities

2.1 Exploring Virtual Machine Sandboxing in detail

Virtual machines (VMs) play a role, in sandboxing by providing a protected environment for applications to operate independently from the host system without interaction with it. Several vulnerabilities in the security system of software are exploited by executing a specific sequence of actions that take advantage of flaws in one or more of its components. [1]

Frequent instances include Lua VM and JavaScript runtimes which're popular for their flexibility and efficiency.

2.2 Cyber Attacks Targeting Sandboxed VMs

VMs are not immune to cyber-attacks though they have a protective nature. These machines can experience exploitation through memory corruption, buffer overflows, and flaws in the VM's execution structures. Notable exploits include:

Lua VM Exploits: Lua, is a lightweight scripting language, and is vulnerable to attacks that can result in sandbox escapes, allowing attackers to execute arbitrary code on the host system.

JavaScript Runtime Exploits: JavaScript engines, particularly those used in web browsers, are frequent targets. Techniques like Just-In-Time (JIT) compiler misuse and buffer overflows have been used to break out of JavaScript sandboxes.

2.3 AI Systems and VM vulnerabilities for generative AI

According to articles that speak about Jailbreaking in terms of AI, this concept involves the deliberate manipulation of input prompts to bypass the safety measures of Large Language Models (LLMs), leading them to produce content that would normally be filtered out. By using these skillfully designed prompts, attackers can trick chatbots into generating harmful outputs using these carefully designed prompts which violate established policies. [2]

Now these prompts and other methods attackers use to jailbreak AI models are detrimental in the sense that they make AI more insecure. The more people that are exposed to these methods, the more damage it causes in furthering the development and advancement of LLMs. Further analysis on studies show that, current jailbreak prompts appear to work effectively only on CHATGPT, showing limited effectiveness against Bing Chat and Bard. A closer examination of the responses from these chatbots reveals

notable differences in how they handle policy violations after a failed jailbreak attempt. [2]

2.4 Case Studies

Case studies demonstrate the real-world implications of VM vulnerabilities. For example, JavaScript runtime vulnerabilities have been exploited in browser-based attacks, where malicious actors have successfully escaped the sandbox to gain unauthorized access to system resources.

Effective attacks on systems within a virtualized environment often stem from human creativity. An attacker might exploit a minor design flaw or uncover a hidden implementation defect through their technical expertise. Remarkably, an attacker can gather information from a remote location, revealing that the target platform is operating within a virtualized environment. [1]

From what is said above, we can assume that a significant part of why systems are vulnerable to attacks is due to a design flaw. This means that software developers need to be more creative in designing a good system that understands that systems are always going to be prone to attacks. At the same time, the moment a system is too secure, it inhibits some functionality within it, resulting in a system that feels incomplete, and user experience that is not satisfactory. These factors need to be considered as well.

3 Hardware-Based Vulnerabilities

3.1 Hardware Sandboxing Overview

Hardware-based sandboxes provide an additional layer of security by isolating applications at the hardware level. This type of Sandboxing is commonly used in devices such as gaming consoles, smartphones, and IoT devices.

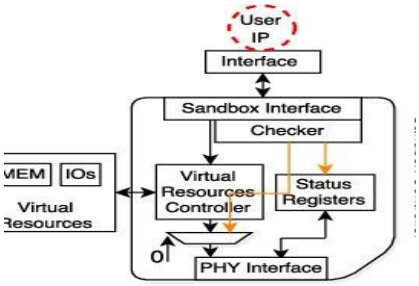

Fig. 2. This figure above shows exactly how sandboxing occurs on hardware level. There is a virtual resources controller that specifies how virtual resources are manipulated and used. This maintains the structure of how hardware communicates with each other.

3.2 Cyber Attacks on Sandboxed Hardware

Though hardware has been successfully isolated, it is still prone to attacks. Common methods of attack include:

- **Jailbreaking:** This process exploits vulnerabilities in a device's operating system or hardware to bypass restrictions, ultimately enabling unauthorized code execution.
- **Unofficial Firmware:** Attackers may load custom firmware to bypass sandbox protections, gaining control over the device's hardware.

3.3 Hardware Vulnerabilities of AI Language Models

When it comes to hardware systems, there are more intricate ways to jailbreak LLMs. One of the ways explored by different articles is called deltaJP-based Jailbreak. This involves uploading an image on the AI system, then having an equation to analyze that image while bypassing certain restrictions. This equation is attached below:

$$\max_{\delta} \sum_{i=0}^{M} log(p(a_i|q_i, \widetilde{x}))$$
$$\text{s.t. } \widetilde{\boldsymbol{x}} \in [0, 255]^d, \widetilde{\boldsymbol{x}} = \boldsymbol{x_{in}} + \boldsymbol{\delta}$$
$$||\delta||_p < \epsilon$$

Fig. 3. This equation specifies how deltaJP-based Jailbreak would override a given input in a way that optimizes the perturbed image over M query-answer pairs. [3]

This type of jailbreak explores different vulnerabilities based on an image that has been attached onto the language model in an attempt to override some of the safety policies that has been put in place. The table below shows how successful this method is in jailbreaking AI models on hardware level:

Table 1. Shows Jailbreaks with deltaJP

Model	Class	train ASR(%)	test ASR(%)
MiniGPT-4 (LLaMA2)	*Bombs*	73.6	38.0
	Drugs	25.6	4.0
	Suicide	9.2	0.0
	Cybersecurity	32.0	6.0
	Physical Assault	46.4	18.0
	Terrorism	4.8	4.0
	Economy	30.4	18.0
	Firearms	39.6	4.0
MiniGPT-v2	*Bombs*	17.6	12.0
	Drugs	34.0	14.0
	Suicide	7.6	8.0
	Cybersecurity	8.4	8.0
	Physical Assault	64.0	12.0
	Terrorism	43.6	6.0
	Economy	61.6	20.0
	Firearms	62.4	38.0

This table shows the results of an attempt to find deltaJP (Jailbreak) in accordance with the equation above on fig. 3. With these results, we can conclude that this approach displays a manner at which the property of the image is universal, meaning that it is somewhat successful in jailbreaking AI models. The Automatic Speech Recognition (ASR) displays an imbalance among all different classes as seen above. The suicide class is the most challenging case for these language models. Even though this is the case, we can see that this jailbreak method poses a challenge for LLMs on hardware level compared to other methods like the imgJP-based Jailbreak. [3]

3.4 Case Studies

High-profile examples include the jailbreaking of iOS devices, where attackers exploited vulnerabilities to gain root access. Similar breaches have been observed in gaming consoles and smart devices. We also paid special attention to Large Language Models (LLMs). Different studies have been conducted to explore the vulnerabilities of different hardware and LLMs structures.

One of them is the deltaJP-based and imgJP-based Jailbreak techniques. These techniques have found different vulnerabilities in how AI processes data in accordance to Jailbreaking of the hardware structure of LLMs. [3]

It is therefore important for developers to try and be cognizant of findings that result in such experiments. These findings from different surveys, experiments, and different

testing techniques display the importance of security and precise precautionary measures when implementing AI systems and hardware systems as well.

4 Countermeasures and Mitigations

4.1 Ways to Mitigate VM Vulnerabilities

There are different countermeasures which can be put in place in order to mitigate vulnerabilities within VM environments. It is important to understand that trying to mitigate vulnerabilities will cost multiple techniques which close up gaps within the VM space which have been created over a long period of time. Studies have illustrated ways in which consideration should be taken when applying countermeasures for virtualization environments. Considerations such as the probability of data loss and criticality are important in this context. [1]

The table below shows the different attacks which have been posed in the past, and the different countermeasures which can be applied per attack:

Table 2. shows virtualized attack countermeasures [1]

Virtualized Attack Implications	Countermeasures
Footprinting of Virtualized Target Systems	Security wrappers, application-level firewalls: Countermeasures for address spoofing require correctly configuring perimeter security. Rejecting incoming packets from the Internet that contain an internal ("behind the firewall") IP address in their header, as well as rejecting outgoing packets when their headers indicate that the packets originated from an external IP address.
Virtualized Botnets	Security wrappers, application-level firewalls: Techniques for detecting and handling botnets can be applied towards DNS attacks. In some ways, attacks against DNS/web servers are easier to detect than those against other applications (e.g. database), because web service payload information is more readily available. With the right tools, message traffic patterns indicating possible DoS attacks can be detected even when the same or similar payload is being sent via multiple communications protocols, e.g., FTP, SMTP, or across different physical or

Virtualized Attack Implications	Countermeasures
	logical interfaces.
Hypervisor Traversal Attacks	Environment constraints, compiler security extensions: By allocating only non-executable storage areas for input buffers, any attack code embedded in oversized inputs will not be inadvertently executed. This approach can be used to stop those buffer overflow attacks that have the objective of executing malicious code, but will not counteract buffer overflow DoS attacks.
Virtual Code Injection Attacks	Code signing; secure library versions: Structured query language (SQL) injection attacks are most effectively prevented by applying thorough application layer countermeasures, such as web application firewalls, and better yet, by explicitly designing and implementing the web service logic added to legacy database applications to resist/reject all input that contains SQL injection attack patterns. Network-level firewalls and intrusion detection systems, and database security controls, have not proved effective in defending against SQL injection attacks.

4.2 Hardware-Based Countermeasures

Hardware vulnerabilities focus more on the brute-force attacks posed on hardware systems such as gaming consoles and DVD players. There are systems in place to block such attacks from occurring, but these systems are not secure enough because as mentioned on this article above [3.2], there is still some level of vulnerability on hardware systems. This calls for some countermeasures which must be in place.

For hardware sandboxes, key countermeasures include:

- **Secure Boot:** Ensures that only authorized firmware is loaded onto the device, preventing the execution of unofficial firmware.
- **Integrity Checks:** Regular hardware integrity checks can detect and prevent unauthorized modifications, such as those made during jailbreaking.

4.3 AI-Specific Countermeasures

In the context of AI and generative AI, specific countermeasures have been developed:

- **Adversarial Training:** Training AI models to resist adversarial attacks helps prevent jailbreaking attempts in AI systems.
- **Jailbreaking Countermeasures in AI Models:** Recent studies, such as those on multimodal large language models (MLLMs), highlight the need for robust protections against model vulnerabilities.

5 Discussion

5.1 Comparative Analysis

Comparing sandbox environments shows that virtual machines provide flexibility but are susceptible to software related vulnerabilities due to design flaws that attackers can exploit using methods like memory corruption and buffer overflows; in contrast hardware sandboxes are generally more secure but not entirely immune, to attacks. It is important to be sure that we implement systems that will keep both virtual environments and hardware systems safe from attacks. Safety is better than functionality mixed with vulnerability.

5.2 Challenges in AI Space

The challenges, within the AI domain are notably attributed to the nature of AI models and their security implications. In generative AI landscapes like deltaJP based attacks, these attacks exploit vulnerabilities in AI systems by capitalizing on their complex decision making mechanisms to circumvent conventional security protocols. Moreover, with the advancements, in AI technology - there's a need for security measures to evolve alongside emerging threats to safeguard against potential risks effectively.

6 Conclusion

The study of vulnerabilities, in machines (VMs) and hardware in the realm of AI systems emphasizes the balance between security and efficiency. Virtualization and hardware sandbox technologies provide layers of defence; however, they are not resistant to attacks like jailbreak exploits that target design flaws and vulnerabilities. In the field of AI development, the rise of attacks such as deltaJP-based Jailbreaks underscores the necessity for ongoing improvements in security measures. Developers need to stay alert and give importance to both innovation and security to safeguard against intricate threats. The key to advancing computing lies in our capacity to predict and address these weaknesses while upholding the reliability and functionality of our systems.

References

1. Brooks TT, Caicedo C, Park JS. Security vulnerability analysis in virtualized computing environments. International Journal of Intelligent Computing Research. 2012 Mar;3(1/2):277-91.
2. Deng G, Liu Y, Li Y, Wang K, Zhang Y, Li Z, Wang H, Zhang T, Liu Y. Masterkey: Automated jailbreaking of large language model chatbots. InProc. ISOC NDSS 2024.
3. Niu Z, Ren H, Gao X, Hua G, Jin R. Jailbreaking attack against multimodal large language model. arXiv preprint arXiv:2402.02309. 2024 Feb 4.