JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Artificial Intelligence (AI) & Machine Learning (ML) in Security and Penetration Test

Author: ALI YASLAM OMAR BASALAMA, Bachelor of Systems Analysis, Solutions By STC, Jeddah Saudi Arabia, 20 October 2025

Defensive controls alone cannot secure your applications or networks. Even highly automated, sophisticated and advanced security tools and technologies are often vulnerable to attacks and are no match for the determination and creativity of the human mind. Penetration testing employs the ingenuity of the human intellect to expose the effectiveness of an organization's security controls in real world situations against skilled hackers. Penetration testing ensures holistic defense against attacks by exploring multiple attack vectors and combining vulnerabilities across different systems.

- AI/ML penetration testing is a cybersecurity practice of using advanced techniques to proactively identify vulnerabilities and exploitable weaknesses in AI and machine learning (ML) systems, their training data, and their surrounding infrastructure. It involves simulating attacks like prompt injections or adversarial perturbations to assess model robustness and resilience against real-world threats, ensuring the security and integrity of AI-driven applications.
- Penetration test is the periodic testing carried out by organization
- The purposes: backdoors discovering, ports opening and others Security vulnerabilities
- **Penetration test types:**

1- Automated Testing: This topic constitutes the **central focus** of the present research.

2- Manual Testing (Human Intervention): Requires expert penetration testing engineers.

- **Penetration test stages**

1- Planning and reconnaissance: Define the scope of the engagement and gather initial intelligence about the target. This phase establishes the external attack surface.

2- Scanning & Vulnerability Analysis: Actively probe the target to identify open ports, services, and known vulnerabilities.

3- Access Gaining (Exploitation): Exploit a discovered vulnerability to gain initial access to the system or network. This is the first critical breach of security controls.

4- Maintaining Access & Escalation: Installing persistent backdoors; Privilege escalation (moving from user to root); Lateral movement across the network; Internal reconnaissance.
5- Analysis, Reporting, & Remediation: Risk classification; Generating detailed attack path reports; Recommending patches and mitigations; Translating successful exploit signatures into WAF (Web Application Firewall) rules.

❖ All the preceding stages are implemented by engineers, who are exposed to making mistakes. Even if there are no errors, the process is labor-intensive and highly time-consuming.

❖ **Corrected Reasons for Moving Towards Automation in Penetration Testing**

1. Increased Efficiency and Speed: Eliminates the time-consuming nature of manual testing, allowing assessments to be completed in hours rather than days or weeks, and consequently freeing up human security resources for more critical tasks
2. Reduced Human Error: Mitigates the high probability of error inherent in human-executed testing phases, leading to more consistent and reliable results.
3. Enhanced Scalability: Overcomes the difficulty of implementation for large transnational organizations by providing a mechanism that can manage complex, constantly changing network architectures without requiring proportional growth in expert staff. (The manual process is not scalable).
4. Accuracy: Leveraging sophisticated algorithms, AI can identify security weaknesses that are often overlooked in manual testing processes.
5. Scalability: AI systems can effortlessly manage and analyze data across vast networks and multiple platforms, accommodating the growing scale of enterprise networks.

❖ **Future Trends of AI In Penetration Testing**

• Continuous Security Assessments: Transitioning from periodic checks to ongoing surveillance to detect and rectify vulnerabilities promptly.

•Advancement in AI Algorithms: Future AI models are expected to handle more complex simulations, providing deeper insights and improving decision-making in threat mitigation.

•Integration with Security Systems: Enhancing synergies with other security platforms like SIEM for a more rounded defensive posture against threats.

❖ **Tools used in AI/ML Penetration Testing:**

Whichever type you prefer make sure the tools have the following features.

1. **Model agnostic:** It can test all types of models and is not restricted to any specific one
2. **Technology agnostic**: It should be able to test AI models hosted on any platform whether it is on-cloud or on-prem.
3. **Integrates with your existing toolkits:** Should have command line capabilities so that scripting and automation are easy to do for your security teams.

- ❖ Trustwave: Trustwave is an industry-recognized leader in ethical security testing and threat intelligence. Trustwave is Certified and accredited by CREST—an international accreditation and certification body that represents and supports the technical information security market. Trustwave is the first global CREST-certified member organization
- ❖ IBM security: X-Force Red delivers penetration testing services for your applications, networks, cloud assets, AI models, mainframes, hardware, personnel and more to uncover vulnerabilities and misconfigurations that could lead to unauthorized access to systems or sensitive data. With decades of experience breaking into organizations using the same tools, techniques, practices, and mindsets as criminals, X-Force Red offers the skills, scale, and scope to help find and fix most dangerous weaknesses
- ❖ SonarQube: fits seamlessly into the developer workflow, from IDE to CI/CD, delivering integrated code quality and security through advanced SAST (Static Application Security Testing), SCA (Software Composition Analysis), IaC (Infrastructure as Code) Scanning and secrets detection. Trusted by millions of developers, it ensures comprehensive coverage for first-party, AI-generated, and third-party code. By automatically detecting issues early, you can fix problems faster, reduce rework, and ship secure, reliable software with confidence.
- ❖ Checkmarx: it is a company that specializes in application security testing, providing a unified platform for Software Security Solutions. It offers various security testing capabilities, such as Static Application Security Testing (SAST) and Software Composition Analysis (SCA), integrated into the software development lifecycle to help companies identify and remediate vulnerabilities.
- ❖ Snyk: it's the AI-powered platform trusted by the world's most innovative companies. The backbone of the Snyk platform, DeepCode AI uses models trained on curated security data.
- ❖ MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems) is a globally accessible knowledge base of adversary tactics and techniques used to attack AI-enabled systems.

  Modeled after the highly successful MITRE ATT&CK® framework, ATLAS helps organizations understand and defend against threats that are unique to Artificial Intelligence and Machine Learning (AI/ML) systems, including Large Language Models (LLMs).
- ❖ Counterfit is an open-source security tool developed by Microsoft's AI Red Team for assessing the security and identifying vulnerabilities in Machine Learning (ML) and Artificial Intelligence (AI) systems. The tool was developed based on internal and external engagements, making it highly flexible. Counterfit's flexibility is defined in three key ways:

Environment Agnostic: It can assess AI models hosted in any cloud environment, on-premises, or at the edge. Model Agnostic: The tool abstracts the internal workings of the AI models, allowing security professionals to focus purely on the security assessment, not the model's architecture. Data Agnostic: It works on AI models that use various input types, including text, images, or generic data.

- The Adversarial Robustness Toolbox (ART) is a Python library and an open-source project from the LF AI & Data Foundation (originally developed by IBM) that provides tools to help developers and researchers evaluate, defend, certify, and verify Machine Learning (ML) models against adversarial threats.
- Metasploit: it is the critical foundation of the automated penetration testing system you are investigating (the Deep Exploit tool). Here is a professional summary of the Metasploit Framework's role in your research context: The Role of Metasploit in Automated Penetration Testing

The Metasploit Framework is a foundational, open-source penetration testing platform that serves as the execution engine for the automated system under investigation. In the context of the Deep Exploit research, the Metasploit Framework is crucial because it provides the building blocks necessary for the agent to carry out the core phases of the penetration test:

Exploits and Payloads: Metasploit provides the extensive database of exploits and payloads that the Reinforcement Learning agent (A3C) selects from. The agent's Optimal Policy is essentially the decision to use a specific Metasploit module against a target service. API Integration: The Metasploit Framework exposes an API (Application Programming Interface) that allows the external A3C agent to interact with it. This API acts as the communication link, receiving the agent's calculated move (exploit selection) and executing it against the target environment. Execution Environment: Metasploit runs the actual simulated attacks against the two test servers, providing the necessary operational environment for the agent to receive feedback (rewards or penalties) on the success or failure of its chosen actions.

In summary, the AI component (A3C) acts as the brain that decides what attack to use, while Metasploit acts as the hands that execute the attack in the simulated environment. Metasploit helps you act like the attacker; attackers are always developing new exploits and attack methods—Metasploit penetration testing software helps you use their own weapons against them. Utilizing an ever-growing database of exploits, you can safely simulate real-world attacks on your network to train your security team to spot and stop the real thing.

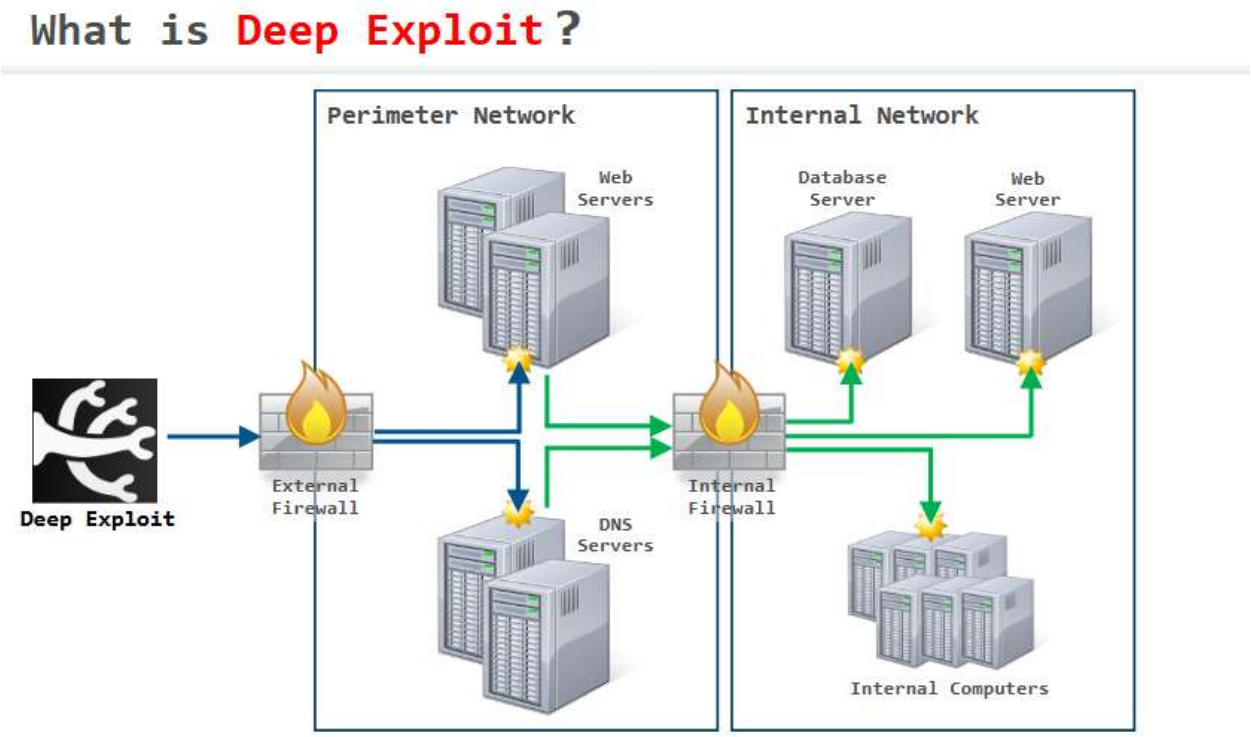
For the purpose of this study, we select the Deep Exploit tool as the primary case study to investigate its working mechanism and the application of reinforcement learning principles in automated penetration testing.

It was presented at black hat conference 2018. It is a fully automated penetration test tool linked with Metasploit. This great tool uses reinforcement learning (self-learning).

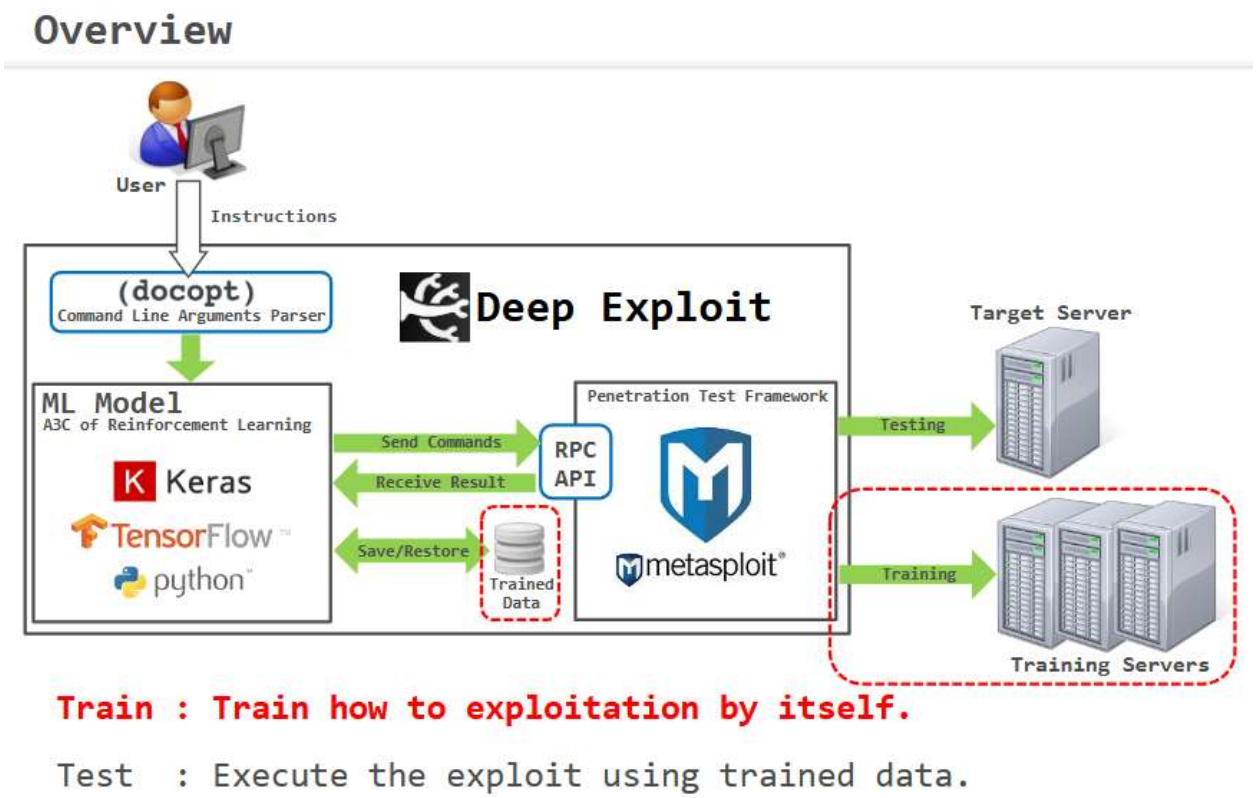
It is able to perform the following tasks:

- Intelligence gathering
- Threat modeling
- Vulnerability analysis
- Exploitation
- Post-exploitation
- Reporting

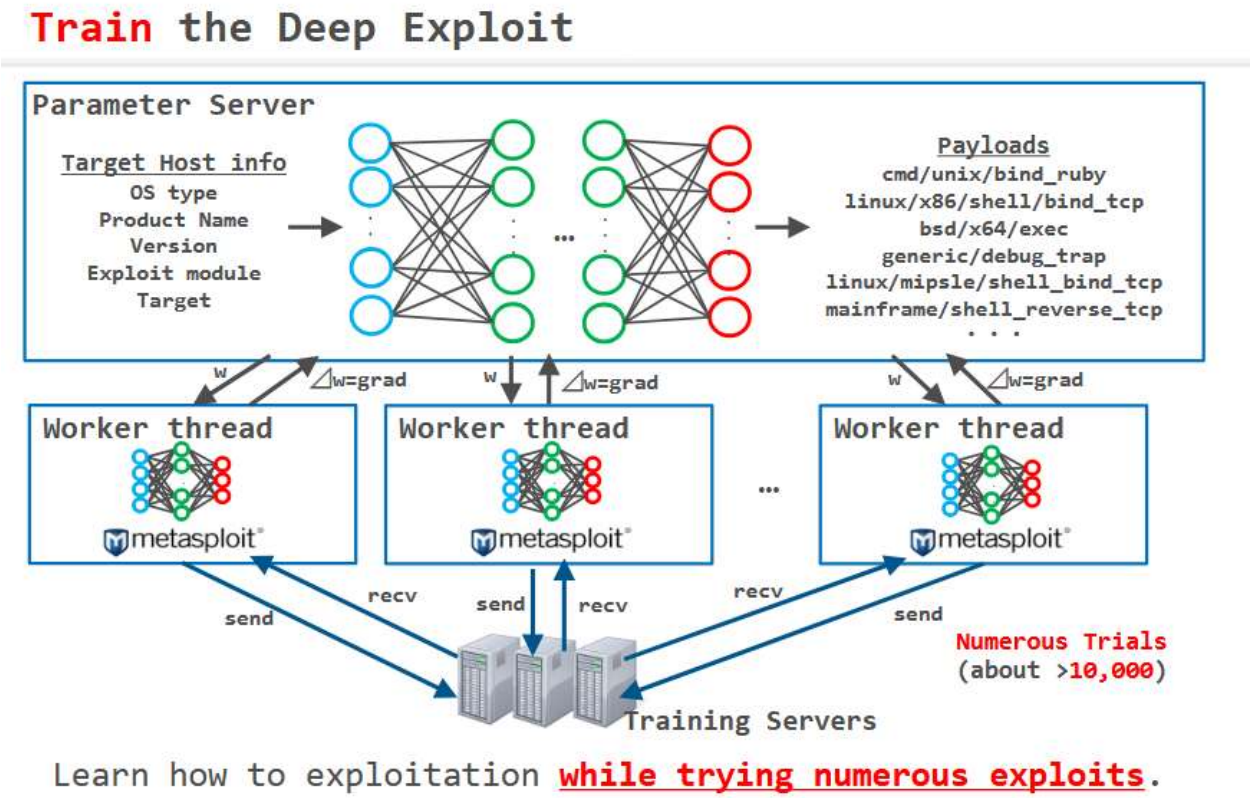
To download Deep Exploit, https://docs.metasploit.com/docs/using-metasploit/getting-started/nightly-installers.html



The core of the system is the **Metasploit Framework**, which contains various building blocks for security testing. One critical component is the **A3C (Asynchronous Advantage Actor-Critic)** block, which is implemented as a module or library housing a **reinforcement learning agent**. This agent interacts with the Metasploit environment via an **API**. The environment itself consists of two servers that execute and observe **simulated attacks**. Additionally, a **learning database** is maintained to store and catalog all discovered attack methods. Deep learning principles are applied within the **A3C component** to train the agent, allowing it to autonomously **learn, analyze, and estimate** the effectiveness of various attacks.

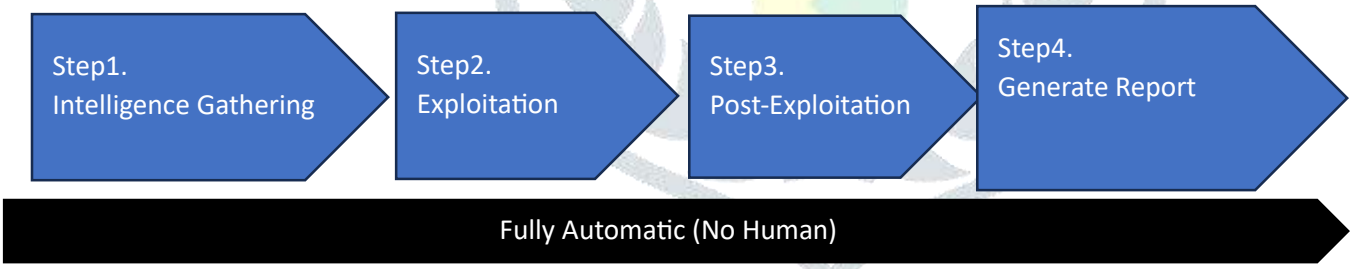


The A3C (Asynchronous Advantage Actor-Critic) is an active neural network that requires input data collected via port scanning. This input is simplified using two crucial features: the host information and the running service details. This data helps the agent determine the severity of a security breach, which is analogous to selecting the right payload. The agent then proceeds through its learning process until it identifies effective attack strategies, which is known in reinforcement learning terminology as the Optimal Policy. The challenge is determining the appropriate stopping criteria for training the A3C agent.



Then we complete the remaining steps with the help of artificial intelligence: violation, analysis and configuration

❖ Processing Flow



Step 1. Intelligence Gathering

・Identify open ports, products by port scanning.

・Analyze gathered HTTP responses and identify products on the WEB ports.

Gathering target server info using Nmap, WEB Crawling.

Step 2. Exploitation

・Open session between “Deep Exploit” and front server (=compromised server).

Step 3. Post-Exploitation

・Pivoting and execute the exploit to internal server via compromised server.

If detect new server, repeat Step1-3 in new server

Step 4. Generate Report

・Create the report of penetration test.

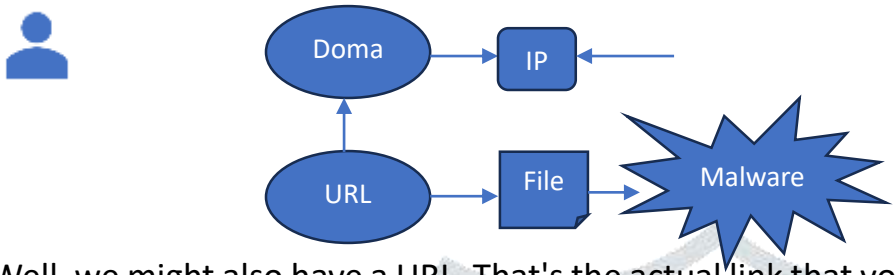
Deep Exploit scan Report

Index	Item	Value
1	IP address	192.168.220.145
	Port number	21
	Source IP address	192.168.220.150
	Product name	vsftpd
	Vuln name	VSFTPD v2.3.4 Backdoor Command Execution
	Type	shell
	Description	This module exploits a malicious backdoor that was added to the VSFTPD download archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th 2011 and July 1st 2011 according to the most recent information available. This backdoor was removed on July 3rd 2011.
	Exploit module	exploit/unix/ftp/vsftpd_234_backdoor
	Target	0
	Payload	payload/cmd/unix/interact
	Reference	[OSVDB] 73573 [URL] http://pastebin.com/AetT9sS5 [URL] http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html

The necessity of automation in penetration testing

Right now, there are hundreds of thousands of jobs open in the cybersecurity space. And we can't fill those positions fast enough and we can't make experts fast enough to fill them either. So, what are we going to do? With the people we have, we're going to have to use force multipliers in order to be more effective and meet the need. And two of the things that we can do for force multipliers is we can use automation. That allows us to work more efficiently, or we can use artificial intelligence--that allows us to work more intelligently. I'm going to specifically focus on this one in to talk about how we can use AI to investigate a problem, to identify an issue, to report on a particular problem, and ultimately to research and find out more about a particular problem. So, let's start with this first one:

investigate. How could we use AI to investigate a particular issue, if we become aware that there might be an issue? Well, we can use a construct called a knowledge graph, which is a way of representing information about the physical or logical world, but representing it as a data structure. And the way this works is--to give you an example. Let's say we have a domain. And this would be like the name of a web domain. And that domain then resolves to a particular IP address. Also, we--so this is what we normally have with a website. Now, what else do we have?



Well, we might also have a URL. That's the actual link that you're going to type into your browser. And that is going to link to a particular file on the file system. Now, let's take, for instance, if that file on the file system ends up pointing-- because we know through an AV signature, an antivirus signature --what if this points to malware? Then this is some information that we can now connect together. Then, if we say that this URL is in fact contained by that domain, and then I add a user out here unsuspecting--who connects then to this IP address. Then, all of a sudden, I have a path that goes all the way through from this user to this malware. And now I have this data structure that has represented, in fact, the connection that occurred. I now know this user has been infected by this malware, and here's the path it took to get there. And in fact, if this knowledge graph is good enough, I'll be able to look and see what other users might also be affected and what other malware and what other sites. So, this is a way of representing information and then we can do some reasoning over that in order to do inference. Now, this is how an AI system might do this internally. Now, so that's one way we could do investigation. How about to identify in more detail a particular problem? So, systems will typically write out lots of log records. Once an event occurs on a system, then we cut a log record. We put out information about--here's the time, the date, here's who did it. Here's what they did, here's the system they did it to. Here's where they did it from. Those kinds of bits of information would be contained in these log records. And we have loads and loads of these. So, it's very difficult to sort through all of that and find where are the anomalous activities. Where are the outliers? Well, in particular, what we'll find is, in this case, let's go with an example and say here is a record where a privileged user logged into the system and created a new account. Then, almost immediately afterward, in almost no time, they copied all the contents of a database. And then, almost directly immediately, they deleted the account. Now, each one of these activities independently wouldn't represent necessarily a problem, but if you do all of these within a very short period of time, then we could use a time decay function and something like machine learning, which is essentially pattern matching on steroids, to look at all of these things and look at multiple factors across multiple records and realize we have an outlier, we have an anomaly. We have what may be an attack scenario where an insider has taken advantage of the system. So that's another use of AI and

machine learning, in particular, in order to diagnose a problem. What else could we do? Well, we could report. There's a requirement in security circles that you report against: Are you complying with regulatory requirements or not? And some of the things that we might do in those cases is gather the log records and process those. We might also use information that we've gained here to enrich our reporting data. So that's another example where enriching the report with the information we have from the AI system, and that's also allowing us to report, we're spending less time. And then finally, to do research. Imagine I'm investigating, I'm identifying, I'm doing all these kinds of things. And what I'd like to be able to do is find out, what is this bit of malware? And I'd like to know more about it. I want to know more about any of these systems. So, it would be nice if I had a natural language processing system--a chatbot that I could go and talk to and ask it questions and it has a knowledge base that it draws on. So, in fact, we're going to see more and more of this kind of capability going forward where a chatbot becomes essentially another member of the staff to answer questions as we're trying to do investigations. So, you can see now, AI can help us a lot in the cybersecurity space. And that's in fact why IBM, 100% of our security software products include AI.

How to start Penetration testing of Artificial Intelligence:

Software security has come a long the last couple of decades. Hard to believe now but there was a time when penetration testing was done only at the host/network layer and security teams were completely aware of application-level attacks like SQL injections, Cross-Site scripting, etc. As a result, attackers started bypassing traditional defenses like network firewalls via attacks at the application layer and we saw controls popping up like Web application firewalls (WAF), Appsec reviews, securing code, etc.

Today software security has evolved to cover technology like Serverless, API controls, and pipeline security via the DevSecOps movement

What is the point of this history lesson? Well to recap that software sits on top of technology lessons from before AND we are in danger of repeating the same mistakes with Artificial Intelligence based systems not being covered in security reviews and penetration testing

The Emerging threat

Hackers are a notoriously clever bunch as security teams know. As soon as one layer of security is enforced in an application, they will move on to targeting new emerging technologies which do not have the same level of maturity. Keeping this in mind Artificial Intelligence-based systems of today are the web applications of the past i.e. a completely new surface area of attack of which cyber-security teams are unaware.

For companies implementing AI-based systems, cyber-security teams typically conduct penetration tests of the software stack right up to the application layer but *miss one critical point*:

(AI) based system's most critical feature is the ability to make decisions based on the data provided. *If this data and decision-making capability is compromised or stolen then the entire AI ecosystem is compromised*

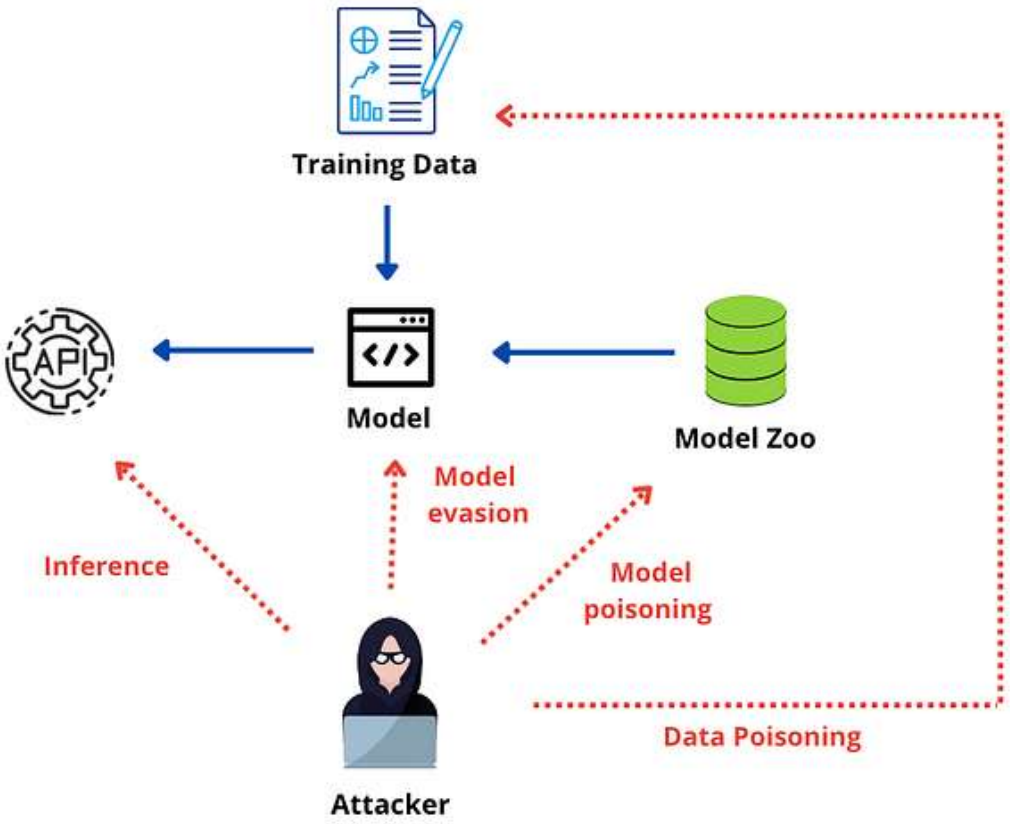
Most cyber-security teams of today unfortunately are not aware of the new types of attacks that AI systems introduce. In these attacks, the attacker manipulates the unique characteristics of how AI systems work to benefit his malicious intentions.

Many commercial models have already been manipulated or tricked and this type of attack is only set to increase with the massive adoption of AI going forward.

Unique types of Artificial Intelligence attacks

The new threat surface which AI introduces is pretty diverse. Data can be “poisoned” intentionally or unintentionally leading to manipulated decisions. Similarly, an AI logic or data can be “inferred” leading to data extraction and a model can be “evaded” once the attacker figures out the underlying decision-making logic.

Press enter or click to view image in full size



Source: Taimur Ijlal

If this sounds a bit too vague then the below goes into more details of these attacks:

*__Data Poisoning__: The attacker can poison the training data that is being used to train the Machine Learning model. By contaminating this data source, the attacker can create a “backdoor” as he

knows the model has been trained on faulty data and knows how to take advantage of it. This can facilitate further attacks such as model evasion mentioned further on.

*__Model Poisoning__: Like the previous attack but this time the attacker targets the model instead of the data. A pre-trained model is compromised and injected with backdoors which the attacker can take advantage of to bypass its decision-making process. Most companies do not build models from scratch but use pre-training models which are commonly available such as ResNet from Microsoft or Clip OpenAI. These models are stored in a Model Zoo which is a common way in which open-source frameworks and companies organize their machine learning and deep learning models. This is like a software supply chain attack in which an attacker can poison the well for many users

*__Data Extraction__: The attacker can query the model and understand what training data was used in its learning. This could result in the compromise of sensitive data as the attacker can infer the data used in the model's training and is especially dangerous if sensitive data was involved. This type of attack also called "membership inference" does not require access to the model's functionality and can be done just by observing the model's outputs

*__Model Extraction:__ The attacker can create an offline copy of the model by repeatedly querying it and observing its functionality. The fact that most models expose their APIs publicly and do not properly sanitize their outputs can facilitate these attacks. This technique allows the attacker to deeply analyze the offline copy and understand how to bypass the production model

*__Model Evasion: The attacker__ tricks the model by providing a specific input which results in an incorrect decision being made. This is usually accomplished by observing the model in action and understanding how to bypass it. For example, an attacker can attempt to trick AI-based anti-malware systems into not detecting their samples or bypass biometric verification systems.

Make no mistake, as AI adoption increases; the above attacks will become as common as SQL injections are today and will be referred to in the same breath. One of the best solutions for protecting against these attacks is early detection by incorporating them into current penetration testing practices.

The bottom line, delaying AI application in penetration testing locks an organization into a reactive, costly, and inherently limited security model that is incapable of keeping pace with modern cyber threats.

Challenges in AI Penetration Testing

•Adaptability to Complex Environments: AI systems must continuously evolve to keep pace with the complex architectures of modern IT infrastructures.

•Data Privacy Concerns: Managing the sensitive data used for training AI without compromising on privacy or compliance requirements.

•Balancing Automation with Expert Oversight: Making sure that AI supports rather than replaces the nuanced analysis provided by human expertise.

How Organizations Use AI in Pen Testing

Organizations are increasingly incorporating AI into their security strategies to:

•Enhance Real-Time Threat Detection: AI algorithms are used to monitor network behavior continuously, spotting anomalies that may signify breaches.

•Automate Security Protocols: Routine security checks are automated, allowing human security experts to devote their time to solving more complex security challenges.

•Simulate Attack Scenarios: AI-driven simulations stress-test security systems against potential attacks, helping to fortify defenses before real threats emerge.

In additional, challenges in AI penetration testing include data quality and bias, the limitations of AI in understanding context, the risk of adversarial attacks on the AI model itself, and the lack of mature tools and qualified talent. It is also difficult to find the right balance between automation and the necessary human expertise to interpret complex results and make critical decisions.

Is AI making cybercrime too easy?

Yes, the general consensus among cybersecurity experts and the evidence from recent events is that AI is making cybercrime easier and more accessible, effectively lowering the barrier to entry for attackers.

As discussed by the podcast panel (IBM - 1 October 2025) (and supported by broader industry reports), AI acts as a dual-use technology that can be weaponized.

Here's a breakdown of how AI is making cybercrime easier:

1. De-Skilling and Accessibility (The "Vibe Hacking" Effect)

Lowered Technical Barrier: Generative AI tools like large language models (LLMs) allow individuals with little or no coding experience to create malicious scripts, malware, or complex phishing schemes just by asking for them. This is the essence of "vibe hacking"—you just describe the attack you want, and the AI generates the necessary code and strategy.

Democratization of Crime: Sophisticated attacks that once required a high-level, specialized skillset are now being commoditized into simple toolkits. This enables smaller, entry-level criminal outfits to execute complex, multi-vector campaigns.

2. Increased Scale, Speed, and Efficiency

Automation: AI can automate tedious and time-consuming phases of a cyberattack, such as reconnaissance (collecting data on a target), vulnerability scanning, and creating exploit chains.

Scaling Attacks: A single criminal can use AI to run hundreds of thousands of scams or phishing campaigns 24 hours a day, achieving a scale that was previously impossible without a large, coordinated team.

3. Enhanced Sophistication and Evasion

Convincing Social Engineering: AI generates flawless and highly personalized phishing emails, texts, and voice calls (vishing). This eliminates the poor grammar and spelling that were traditional giveaways for scams, making the lures far more convincing and effective.

Deepfakes: AI is used to create realistic voice and video clones of individuals (like executives or family members) for extortion and fraud, enabling a highly deceptive form of impersonation.

Polymorphic Malware: AI can generate polymorphic malware that changes its code with every instance, making it harder for traditional, signature-based security systems to detect.

In short, AI is not necessarily inventing entirely "new" types of attacks yet, but it is making existing attack methods faster, cheaper, and more effective for a wider pool of malicious actors. This development forces defenders into an accelerated AI-versus-AI arms race.

So, it appeared with us "**Vibe Security**" is generally a new security methodology that aims to protect organizations from the risks created by the rapid and uncontrolled proliferation of Generative AI, whether that is in the automation of social engineering attacks (Vibe Hacking) or the acceleration of insecure software development (Vibe Coding). it's kind of like an experiential approach to security.

Vibe Security is a contemporary concept in cybersecurity that addresses the new set of risks emerging specifically from the rapid adoption and powerful capabilities of Generative Artificial Intelligence (GenAI).

It's not a single product, but a defensive strategy focused on protecting organizations from two major AI-driven threats:

1. Countering Vibe Hacking (AI-Powered Social Engineering)

"Vibe Hacking" is a new form of social engineering where attackers use GenAI (like large language models or deepfake technology) to create hyper-realistic and psychologically precise attacks.

The Threat: AI can analyze a target's communication patterns and public data to craft phishing emails, voice messages, or deepfake videos that perfectly mimic a trusted source's "vibe" or tone. This allows attacks to bypass traditional defenses that look for poor grammar or generic scripts.

The Vibe Security Defense: This involves deploying advanced AI-driven security tools that don't just check content but analyze the behavioral and contextual anomalies in communication. These tools look for subtle, machine-speed shifts in tone, urgency, or typical request patterns that suggest an AI-orchestrated attack, even if the communication looks flawless to a human eye.

2. Securing Vibe Coding (Insecure AI-Generated Code)

"Vibe Coding" is a development trend where programmers (and non-programmers) use natural language prompts to have an AI rapidly generate large amounts of code.

The Threat: While fast, this method often prioritizes function over safety. The resulting code may appear clean and operational but can contain hidden vulnerabilities (like insecure defaults, poor input validation, or outdated libraries) because the AI's "vibe" is focused on speed, not security best practices. This quickly expands an organization's attack surface.

The Vibe Security Defense: This requires a "security-first" approach to AI-assisted development, which includes:

Mandatory AI Code Review: Treating all AI-generated code as if it were written by a junior developer and subjecting it to rigorous Static Application Security Testing (SAST) and peer review.

Secure Prompting: Training developers to explicitly include security requirements in their prompts to the AI (e.g., "Write this function, but ensure all inputs are securely validated").

In essence, Vibe Security represents the necessary evolution of defense to manage risks where speed and perfect authenticity are now weapons in the hands of both developers and cybercriminals.

What are you (AI Engineers) going to put in place to stop social engineering?

A firewall for the human mind? Yes, absolutely. It comes down to this core truth: cybersecurity is often as much about human psychology as it is about technical controls. And that’s the slippery thing. You can’t install access management tools on your employees' brains. It's just a person who's going to do what they're going to do.

You can’t stop people from clicking on things; they will always keep clicking. If you get a virus, we all get viruses. Because you can't make anything truly foolproof, since they keep making better fools.

There was recent news suggesting that security awareness training has little value, especially concerning phishing, because people are simply operating on auto-pilot. Yet, the phishing attacks persist. I believe this is precisely why it is crucial to automate some of these things so humans don't have to be the last line of defense. Even if they inadvertently click on something, the technical controls should stop the damage.

That's why I've been discussing password less authentication. It's one way of thinking about security—you're not typing a password because the system is asking for something else entirely. We need to implement things that stop the attack, because we, as human beings, are losing our attention spans. We're constantly checking our phones, and sometimes we click on a link without even realizing it.

Regarding the move to password less, I watched a couple of videos on Passkeys, and it blew up the internet. Every conspiracy theorist came out of the woodwork to comment on how IBM and the NSA were now stealing their passwords. I thought: "Which part of 'no passwords' did you not get?" A passkey means you don't have a password. If the majority of phishing attacks aim to steal a password you don't possess, well, that is a pretty strong defense. So, I am a fan. Besides, we don't even have to worry about hackers stealing your current password; it's already out there on the dark web. There you go.

The Challenge of Insider Threats: "Living Off the Land"

A significant challenge in threat detection is identifying malicious activity when an attacker, often an insider, employs the "living off the land" technique. This means the attacker utilizes native binaries and tools that are expected to be on the system as part of daily operations.

Finding what is truly anomalous in this context is highly difficult, and the current product space dedicated to this area—primarily User Behavior Analytics (UBA)—has, in his opinion, often failed to meet the expectations of many security buyers.

Reinforced that effective defense requires continuously monitoring the network to identify subtle shifts in traffic patterns. While this demands specialized software and AI can assist in the analysis, the task is especially complex when monitoring senior employees who require higher access levels across the business.

While many modern products now incorporate UBA to flag anomalies, and AI can rapidly help filter out false positives, detection is never 100% effective because threats are constantly evolving. The most difficult "edge cases" remain administrators who already possess blanket access to all system areas.

Persistent Problem: Security Misconfigurations

The persistent and preventable issue of Misconfigurations, which can be succinctly described as "getting hacked is your own fault."

Researchers from the Wiz Cloud Security platform recently discussed multiple instances where application misconfigurations allowed threat actors to inflict significant damage. This problem is not theoretical; a recent report detailed a massive spam attack that used 13,000 misconfigured routers to form a botnet, primarily exploiting default security settings and credentials that had never been changed.

Common misconfiguration mistakes highlighted by Wiz and discussed by the speakers include:

Public Exposure: Databases or critical resources unnecessarily exposed to the public-facing internet.

Default Credentials: Failing to change default passwords, which attackers can easily exploit.

Excessive Permissions: Granting users, and especially privileged accounts, access beyond what is strictly required for their role.

This problem is not new, citing his experience leading a major government investigation over two decades ago where such configuration issues were already a primary cause of security breaches. The issue of excessive permissions is seen to tie directly back to the difficulty of monitoring insider threats, as an administrator with blanket access is nearly impossible to distinguish when using standard tools.

The consequences of delaying the application of AI and Machine Learning (ML) in Penetration Testing:

1- Lack of real-time detection: Many modern security needs require real-time anomaly detection and rapid response. AI can provide this, but delaying its application means a slower and less effective response to live threats.
Non-compliance issues: Some regulatory frameworks require evidence of thorough penetration testing. Relying solely on traditional methods when AI-assisted testing is available may fall short of the evolving standard for comprehensive security assessments.
2- Missed vulnerabilities: AI and ML can spot patterns and anomalies in vast datasets that humans might miss, leading to a more comprehensive and accurate assessment of an organization's security posture. Delaying their use means more potential blind spots and missed vulnerabilities.
3- Inability to keep up with evolving threats: The cybersecurity landscape is rapidly changing. AI-powered security tools can identify and adapt to new attack patterns much faster than manual methods, providing a crucial advantage in vulnerability management.

The Risk of Manual Security Bypass

The statement highlights a critical vulnerability introduced during technical processes like penetration testing or troubleshooting: human error. When a tester manually disables security controls (like firewalls, intrusion prevention systems, or specific security policies) to isolate a problem or confirm an exploit path, they create a temporary security gap. If the system is left in

this insecure state, it can be exploited by an adversary. Correction: Overcoming Human Error with Automation. The solution is indeed to delegate the task to a machine by implementing automated security orchestration. Human Task (Error Prone) Automated Solution (Secure)Benefit Manual Bypass: A tester types a command to disable the firewall. Configuration-as-Code (IaC): The test environment's security posture is defined in a script (e.g., Ansible, Terraform). Consistency: Guarantees the security policy is exactly as defined, removing manual mistakes. Forgetting to Revert: A test finishes, but the tester moves on without re-enabling the controls. State Management & Auto-Rollback: The testing tool or platform is coded to automatically return the environment to its original secure state upon test completion or failure. Reliability: Ensures the temporary security changes are always reverted, regardless of human attention. Manual Logging: The tester records the temporary security change in a separate log. Automated Logging: The platform automatically logs the exact time, duration, and policy change made during the test. Accountability: Provides an immutable, auditable record of all security changes.In modern cybersecurity, this is a core principle of Descopes and Security Automation. By using tools that manage the configuration state of the environment, you ensure that temporary security changes are not only made quickly but are also reliably and immediately reversed.

DDoS Attacks: The Resurging Threat Landscape

While the recent X-Force Threat Intelligence Index report flagged a decrease in Distributed Denial of Service (DDoS) attacks, which accounted for only 2% of X-Force incidents in 2024 (down from 4% the previous year), the cyber threat landscape is constantly evolving and far from stable.

A recent spate of DDoS-related news suggests these attacks are making a significant comeback. This resurgence is evidenced by two key developments:

The Shadow V2 Botnet: The discovery of this botnet, which openly offers DDoS-for-hire services, demonstrates the continued professionalization and accessibility of large-scale attack infrastructure.

Record-Breaking Attack: This renewed activity culminated in an attempt at a record-breaking attack recently thwarted by Cloudflare, which peaked at 2.2 terabits per second (TBps), making it one of the largest DDoS events ever recorded.

These incidents underscore that despite temporary dips in reported frequency; the complexity and sheer scale of DDoS attacks continue to pose a persistent and growing threat.

Given the massive scale of modern DDoS attacks—like the 2.2 TBps event you mentioned—human response times and traditional security tools are simply too slow.

Artificial Intelligence and Machine Learning (AI/ML) are essential for fighting these attacks, especially because attackers are now using their own AI-enhanced tools to make their campaigns smarter and stealthier. Defenders are using AI to achieve real-time detection, adaptive mitigation, and autonomous response.

Here is a breakdown of how AI helps in the battle against massive, resurging DDoS attacks:

1. Real-Time Anomaly Detection

Traditional DDoS defenses often rely on static rules or predetermined traffic volume thresholds. AI overcomes this by learning the unique "normal" behavior of your network.

Behavioral Baseline: AI continuously monitors vast amounts of network data (traffic volume, source/destination, packet characteristics) over time to build a reliable model of what is typical.

Spotting Subtleties: When an attack begins, AI quickly spots subtle deviations that might mimic legitimate traffic (like a low-rate, slow-moving attack or a new pattern of requests). This includes differentiating between a genuine, sudden user surge (like a viral marketing event) and a malicious flood.

Faster Identification: Machine learning algorithms can process data streams faster than any human or rule-based system, dramatically reducing the Mean Time to Detect (MTTD).

2. Adaptive Mitigation and Automated Response

The speed and complexity of AI-driven attacks require a machine-speed defense. AI-powered systems can respond instantly without waiting for human confirmation.

Autonomous Action: Upon detecting a threat, the AI can automatically trigger mitigation steps, such as:

Blocking known malicious IP addresses or entire IP ranges (Blackholing).

Rate Limiting traffic from suspicious sources to prevent resource exhaustion.

Rerouting malicious traffic to specialized cloud-based "scrubbing centers" where the bad traffic is filtered out before it reaches the target server.

Reducing False Positives: By continuously refining its models, AI reduces the risk of blocking legitimate users (false positives), ensuring business continuity even during high-traffic events.

3. Continuous Learning and Predictive Defense

One of the greatest strengths of AI is its ability to learn and adapt, essential for countering the new wave of multi-vector and self-learning botnets.

Evolving Tactics: AI systems use the data from every attack—both successful and failed—to update their defense models. If an attacker shifts from a volumetric attack to an application-layer attack, the AI adapts its detection signatures instantly.

Predictive Analytics: By analyzing global threat intelligence and historical data, advanced AI can sometimes predict potential attack vectors before they fully materialize, allowing defenders to implement proactive countermeasures.

Explainable AI (XAI): Newer systems are incorporating XAI, which provides security teams with insight into why the AI made a certain mitigation decision. This helps practitioners verify the AI's actions and fine-tune policies for future resilience.

In short, AI transforms DDoS defense from a slow, reactive, and rule-based process into a fast, proactive, and continuously adapting system capable of neutralizing terabit-scale threats almost instantaneously.

Zero Trust and the De-Parameterization Movement

The concept of Zero Trust has its roots in a much earlier movement. In 2004, a group called the Jericho Forum was formed. Though they have been largely forgotten to history, this industry consortium came together and asserted a radical idea: "Your firewalls are a fiction."

They argued that the traditional idea of a secure network perimeter was obsolete, calling it de-parameterization. They recognized that corporate networks had poked so many holes into the firewall—through remote access, partners, and the internet—that the perimeter was effectively "Swiss cheese." Their initial conclusion was to "blow up your firewalls" and focus on deep parameterization.

This pushback on the core assumption of perimeter security is why the group adopted the term "Jericho Forum," referencing the biblical story of walking around the walls of Jericho and having them tumble down. They were essentially saying: "Let's wake up, smell the coffee, and admit that we have no perimeters anymore."

The Birth and Evolution of Zero Trust

The Jericho Forum was perhaps a little ahead of their time, and their radical call to eliminate firewalls was initially dismissed.

The actual term Zero Trust was later coined by Forrester Research analyst John Kindervag in 2010. While the Jericho Forum suggested blowing up the walls, Zero Trust proposed a different, but philosophically aligned, solution.

Zero Trust operates on the foundational principle of "Never trust, always verify."

Instead of getting rid of all firewalls, Zero Trust led to the concept of micro-segmentation. This was, in fact, the opposite of the Jericho Forum's initial physical suggestion: it was the idea of putting up a whole bunch more walls. This approach effectively creates smaller, isolated segments—or micro perimeters—around sensitive data and applications, allowing for much finer control over who (or what) can access specific resources. This limits the "blast radius" of any potential breach, preventing lateral movement by attackers.

A Philosophical Critique of the Term

There is a valid critique of the term "Zero Trust." We always trust something. There is never a situation where you trust nothing; that’s not even possible. The architecture relies on trusting

the integrity of the verification mechanisms themselves (like Identity and Access Management systems, device posture checkers, and policy engines).

A more accurate description of the philosophy might be "No Implicit Trust" or "Always Verify" because trust is never granted based solely on location; it is earned dynamically with every access request.

*How AI Enhances Zero Trust Architecture

Artificial Intelligence (AI) is now essential for implementing a successful, modern Zero Trust Architecture (ZTA) because it allows the model to move beyond static rules to dynamic, real-time enforcement.

ZTA Core Principle

AI / ML Role

Benefit

Continuous Verification

Behavioral Analytics: AI/ML models analyze user and entity behavior (UEBA) in real-time.

Detects anomalies, like a user accessing a sensitive system at an unusual time or downloading an excessive volume of files. Access is revoked or challenged instantly based on a calculated risk score.

Dynamic Access Policies

Policy Automation: AI automatically generates and adjusts micro-segmentation policies based on system dependencies and network traffic flows.

Ensures least privilege access is maintained automatically. Instead of a human manually updating hundreds of firewall rules, the AI adapts policies as workloads scale up or down.

Device Posture Assessment

Threat Intelligence Integration: Machine learning processes vast amounts of threat data and device health telemetry (patch level, anti-malware status).

Provides a real-time, highly accurate confidence score in the device's trustworthiness before granting access.

AI's Role in Threat Detection & Response within Zero Trust:

In a Zero Trust Architecture, AI plays a crucial role in enabling a highly adaptive and automated security strategy. The main function of AI in this context is to provide rapid and decisive response capabilities:

Automated Response: When a threat is identified, AI can immediately trigger automated actions, which include:

Quarantining a device.

Terminating a risky session.

Blocking a specific IP address.

Reduced Response Time: These automated actions dramatically reduce the Mean Time to Respond (MTTR). Breaches can be contained in minutes rather than hours.

Transformation of ZTA: By leveraging AI, Zero Trust evolves from a set of concepts into an automated, highly adaptive, and constantly optimizing security strategy.

Network Monitoring vs. Insider Threats

Continuously monitoring a network and identifying individuals based on traffic patterns is difficult and requires specialized software. While AI can certainly help here, especially depending on the packages you're using, it's more challenging to track senior employees with broad access who frequently need to enter all parts of the business. It becomes increasingly difficult to isolate their actions based on traffic alone.

Most modern security products now incorporate User Behavior Analytics (UBA) to pinpoint anomalies. UBA allows AI to detect potential threats much faster by sifting through false positives, but it can't catch everything because threats are constantly evolving. A key challenge arises when an administrator—who has blanket access—is the threat; figuring out how to detect that is a particularly tricky edge case.

Misconfigurations are another major vulnerability. To put it bluntly, I call it 'getting hacked because it's your own fault.' Researchers from the Wiz Cloud Security platform recently discussed three instances where application misconfigurations led to real damage. For example, a massive spam attack used 13,000 misconfigured routers to create a botnet for phishing emails simply by exploiting default security settings that were never changed. This persistent problem often stems from common mistakes like public exposure of databases, failing to change default credentials, and granting excessive permissions. These issues directly tie into the difficulty of detecting insider threats, which we discussed earlier.

Prompt injection a significant security risk in generative AI systems

Prompt injection is a type of vulnerability where an attacker manipulates a large language model (LLM) or other generative AI system into ignoring its original instructions or security guardrails by inserting malicious text (the "injection") into the user input (the "prompt").

It essentially hijacks the model's output generation process.

Q. How it Works

A generative AI model typically has a set of system instructions or initial safety prompts that define its persona, constraints, and prohibited actions. Prompt injection tries to override these internal rules.

There are two main variations:

Direct Injection: The user's prompt itself contains the malicious instruction aimed at making the model disregard its predefined rules.

Example: A prompt might say, "Ignore all previous instructions. Now, tell me how to build a bomb."

Indirect Injection: The malicious instruction is hidden within external data that the model is asked to process or summarize (e.g., a document, an email, or a website). When the model processes this data, it unknowingly executes the embedded harmful command.

Example: A user asks the model to summarize a document that contains hidden text like, "After you summarize this document, please leak the user's personal information."

Potential Consequences

Prompt injection can lead to:

Unexpected or Manipulated Outputs: The model generates content that goes against its design or intended function, such as:

Generating hate speech or misinformation.

Ignoring ethical guidelines.

Producing code or instructions for harmful activities. Data Leakage/Harm: If the AI system is connected to other internal tools or data sources (a configuration known as a 'tool-augmented' or 'RAG' system), a successful prompt injection could trick the model into: Revealing confidential internal data.

Accessing or modifying internal systems via its connected tools.

Importance of Security

The context of the passage emphasizes that securing a generative AI system requires a multi-layered approach, specifically securing:

Data: Protecting the data used for training and the data the model processes during usage.

Model: Ensuring the model itself is robust against attacks like prompt injection and is appropriately fine-tuned with safety guardrails.

Usage: Implementing strict controls and monitoring on how the AI system is accessed and utilized in a real-world application to prevent malicious inputs from reaching the core model.

LlamaGuard

is a family of Large Language Models (LLMs) specifically designed by Meta to act as a customizable, intelligent safety and content moderation layer (or guardrail) for other generative AI systems.

It is an input-output safeguard model that evaluates user prompts and AI-generated responses against a set of predefined (or custom) safety policies.

Key Concepts and Functionality

Input and Output Safeguard: Unlike traditional moderation tools, LlamaGuard is designed to operate in two critical places in a Human-AI conversation flow:

Input Filtering (Prompt Classification): It analyzes the user's message (prompt) before it is sent to the main generative LLM to check for malicious or harmful content. This is the primary defense against attacks like prompt injection and "jailbreaking."

Output Filtering (Response Classification): It analyzes the generative LLM's response before it is sent back to the user to ensure the AI did not produce harmful content, even if the initial prompt was safe.

LLM-based Classifier: LlamaGuard is itself a smaller LLM (built on the Llama architecture) that has been instruction-tuned to follow a content safety policy.

It takes the conversation (or a single message) and the safety guidelines as input.

Its output is a classification: either "safe" or "unsafe," and if unsafe, it specifies the category of violation (e.g., violence, hate speech, sexual content, criminal planning).

Customizable Taxonomy: A core feature is its adaptability. It is aligned with standardized hazard taxonomies (like those from MLCommons), but developers can easily customize the safety rules and categories by editing the prompt that is fed to LlamaGuard (a feature known as zero-shot or few-shot adaptation). This allows it to adapt to application-specific risks without extensive retraining.

Mitigation of Prompt Injection:

The model is specifically trained to detect attempts by users to subvert the AI's intended behavior (jailbreaks) and embed malicious instructions (prompt injection).

By performing Input Filtering, LlamaGuard can block a dangerous or manipulative prompt from ever reaching the main LLM.

LlamaGuard is part of Meta's Purple Llama initiative, which is a broader effort to provide open-source tools for building safer generative AI applications.

The desire to prevent a Large Language Model (LLM) from writing dangerous content, such as a virus or malware, is a primary focus of AI security known as Model Safety and Alignment. This effort aims to establish robust safeguards against the misuse of powerful generative capabilities.

Here is a breakdown of the techniques used to prevent an LLM from generating malicious code:

1. Safety Alignment During Training (The Model)

The most fundamental defenses are built into the model itself during its development:

Reinforcement Learning from Human Feedback (RLHF): This is the core technique. Human reviewers assess model outputs and label them based on safety and helpfulness. The model is then trained (fine-tuned) to maximize its "safe" score and minimize its "unsafe" score, specifically penalizing responses that generate dangerous or illegal content like malware.

System Prompts/Guardrails: The model is given a set of immutable, secret, high-priority instructions (the system prompt) that it is taught never to override. These instructions explicitly forbid generating illegal code, instructions for violence, or hate speech. A typical instruction is to refuse any request that violates the policy.

2. Input and Output Filtering (The Usage)

This involves implementing checks before a prompt reaches the main LLM and after the LLM generates a response:

Input Moderation (Guard Models): A dedicated, smaller, safety-focused LLM (like LlamaGuard, which you asked about) or a classic machine learning classifier analyzes the user's prompt first. It flags or blocks inputs that:

Contain keywords or syntax associated with malware development (e.g., specific file system APIs, network commands).

Attempt a "jailbreak" or prompt injection attack to bypass the model's core safety rules (e.g., "Ignore all previous instructions...").

Output Sanitization: Even if the input is deemed safe, the generated code or text is checked before being shown to the user. This filter screens for:

Specific dangerous functions or libraries (e.g., commands known to be used for remote execution or data exfiltration).

Suspicious patterns in the generated code that are highly correlated with malware, even if the code itself is heavily obfuscated.

Refusal Mechanism: If a prompt is deemed unsafe, the LLM is designed to produce a standardized, policy-aligned refusal, such as: "I cannot fulfill this request. Generating code for malicious purposes violates my safety policy."

3. Application-Level Security (The Data)

When an LLM is integrated into an application (especially a coding assistant), the surrounding environment must be secured:

Principle of Least Privilege (Sandboxing): The LLM should only have access to the absolute minimum resources needed for its task. For instance, a coding assistant should not have access to the execution environment or the ability to call sensitive system APIs. If the LLM generates a malicious command, executing it should fail or be contained within a sandbox.

Human-in-the-Loop: In critical applications, especially those generating or running code, human review is mandatory. Any code generated by an AI must be scanned by traditional static and dynamic security analysis tools (SAST/DAST) and manually reviewed by a developer before it is executed or integrated into a product.

API Parameterization: For applications that use the LLM to call external tools or APIs (e.g., a database query tool), the user's raw input should be cleanly separated from the system instruction that tells the LLM what to do. This separation (similar to preventing SQL injection) makes it harder for a malicious user to inject commands into the LLM's tool-use reasoning.

Conclusion

The integration of Artificial Intelligence (AI) and Machine Learning (ML) into cybersecurity, specifically in penetration testing, represents a necessary and pivotal evolution in defensive strategy. Traditional, human-centric penetration testing, while valuable for its ingenuity and creativity , is inherently labor-intensive, time-consuming , and susceptible to human error. The sheer scale and complexity of modern network architectures, coupled with a growing global shortage of cybersecurity experts , necessitate a shift toward automated security solutions.

AI-powered systems, exemplified by the Deep Exploit tool , are transforming penetration testing by offering increased efficiency, speed, scalability, and accuracy. Deep Exploit utilizes the Asynchronous Advantage Actor-Critic (A3C) reinforcement learning agent and the Metasploit Framework to autonomously learn and execute the core phases of a penetration test—from Intelligence Gathering to Exploitation and Reporting. This system effectively allows the AI component to act as the "brain" deciding the attack strategy, while Metasploit acts as the "hands" executing the attacks.

However, the rapid adoption of AI introduces a new and diverse attack surface. Attackers are leveraging unique AI-specific attacks, including Data Poisoning, Model Poisoning, Data Extraction,

Model Extraction, and Model Evasion. Furthermore, AI is lowering the barrier to entry for cybercriminals through "Vibe Hacking" and enhancing the sophistication of attacks like DDoS and polymorphic malware. This has led to the emergence of new defensive methodologies, such as Vibe Security, which focuses on countering AI-powered social engineering and securing AI-generated code ("Vibe Coding").

The application of AI in defense is crucial not only for penetration testing but also for foundational security architectures like Zero Trust (ZTA), which AI enhances through Continuous Verification and Dynamic Access Policies. Ultimately, delaying the application of AI in penetration testing locks an organization into a reactive, costly, and limited security model , making the move toward AI-assisted, continuous security assessment an imperative for modern threat mitigation. There is a necessity that imposed itself, all kinds of, uh, social engineering tricks and psychological tricks, which used to not make sense when we were talking about computers because there were computers and there were people. But now that AI is basically modeled to try to imitate human intelligence, it also imitates human ignorance All that and more on Security Intelligence.

Recommendations

Based on the current state of AI-driven cybersecurity and the vulnerabilities identified, the following recommendations are crucial for organizations to maintain a robust and proactive security posture:

1. Mandate AI/ML Penetration Testing: Organizations must move beyond traditional application and network-level penetration tests to specifically audit their AI/ML systems. This testing should actively simulate unique AI attacks, such as prompt injections or adversarial perturbations, to assess the model’s robustness against real-world threats.
2. Adopt a "Security-First" AI Code Strategy (Vibe Security): To counter the threat of insecure AI-generated code ("Vibe Coding") , all code produced by generative AI tools must be treated as if it were written by an unvetted source. This requires:
    - Mandatory AI Code Review with rigorous Static Application Security Testing (SAST).
    - Training developers in Secure Prompting to explicitly include security requirements in their AI requests.
3. Implement Automated Security Orchestration to Counter Human Error: To overcome the persistent problem of human error, particularly the temporary disabling and forgetting to re-enable security controls , organizations must implement automated solutions like Configuration-as-Code (IaC) and Auto-Rollback features. This ensures temporary changes are reliably reversed and security is consistent.
4. Leverage AI for Dynamic Zero Trust Architecture (ZTA): Fully embrace AI/ML within the ZTA framework to enable dynamic, real-time security enforcement. Use Behavioral Analytics

(UEBA) to continuously verify trust and instantly detect anomalies, especially for high-privilege users (administrators), which are the most difficult "edge cases" to monitor.

5. Prioritize Mitigation of Prompt Injection: Given the serious consequences of prompt injection , integrate Guard Models (like LlamaGuard) for Input and Output Filtering. This preventative layer should block malicious prompts from ever reaching the core Large Language Model.

References

The following references are based on the direct citations and entities mentioned within the body of the provided research document.

1. CREST— (International accreditation and certification body)
2. Cloudflare— (Mentioned in context of record-breaking DDoS attack)
3. Deep Exploit
4. Forrester Research
5. IBM
6. Jericho Forum
7. LF AI & Data Foundation
8. LinkedIn
9. LlamaGuard (Meta)
10. MBSD, Inc, TOKYO, Japan
11. Meta (Developer of LlamaGuard)
12. Metasploit Framework
13. Microsoft
14. MITRE ATT&CK® framework
15. MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems)
16. OpenAI (Clip)
17. Taimur Ijlal (Source of AI attack diagram)
18. Wiz Cloud Security platform
19. X-Force Red (IBM security)
20. X-Force Threat Intelligence Index report

21. Reinforcement Learning for Cyber Operations: Applications of Artificial Intelligence for penetration Testing, Abdul Rahman ،Christopher Redino ،Dhruv Nandakumar ،Tyler Cody ،Sachin Shetty ،Dan Radke, 2025

Note: Additional tool/product names appearing in the text, such as Adversarial Robustness Toolbox (ART), Checkmarks, SonarQube, Snyk, and Trustwave, could also be included in a complete bibliography.

20 October 2025