Immunefi

CHATGPT SECURITY REPORT

PREPARED BY IMMUNEFI

1. OVERVIEW

ChatGPT Security Report

PREPARED BY IMMUNEFI

The team at Immunefi, the leading bug bounty and security services platform for web3 which protects over $60 billion in user funds, releases the **ChatGPT Security Report**, a comprehensive overview of the current level of adoption and main use cases of the technology among the web3 security community, and a description of ChatGPT-generated bug reports.

Home to the largest community of security talent in the crypto space, Immunefi maps:

- The use, general sentiment, and level of satisfaction towards the technology
- Security use cases
- Limitations and challenges when using the technology
- Level of confidence in ChatGPT vulnerability discovery
- Frequency of use of the technology
- Recommendation of the technology
- Security concerns and main threats the technology poses

ChatGPT Security Report

OVERVIEW

On November 30, 2022, OpenAI publicly released ChatGPT, an artificial intelligence chatbot capable of generating human-like text. ChatGPT quickly took the world by storm, crossing over 1 million users in 5 days and becoming the fastest-growing app in the world after reaching 100 million users within 2 months of its launch. Its impressive capabilities were showcased in writing essays, supporting studying and research, and much more. However, and just as quickly, the use of ChatGPT raised concerns about privacy, security, and the ethical use or misuse of the chatbot.

As an AI model that learns from vast amounts of data, there are risks associated with biased or inappropriate responses, potential misuse for spreading misinformation, and the need for robust security measures to protect against unauthorized access and data breaches. Such concerns sparked a successive ban on the technology across several companies aiming to safeguard confidential data, as employees could disclose trade secrets or client information when interacting with the AI system. According to a report, the list of companies that have banned or limited the use of ChatGPT include Apple, JPMorgan Chase, Deutsche Bank, Verizon, Northrop Grumman, Samsung, Amazon, and Accenture.

While traditional corporations grappled with the technology, a discussion on use cases and concerns also emerged in the web3 industry. The community started to discuss the potential of ChatGPT for smart contact development, testing, security, and a possible increase in security breaches.

But as the community engages further with the technology, where does ChatGPT stand when it comes to web3 security?

ChatGPT Security Report

THE SPAM INCREASE AND THE CHATGPT BAN

After ChatGPT was released, Immunefi started to receive a flood of bug reports that were very well-written, properly formatted, and used the same technical language commonly seen in successful bug reports. But upon further examination, it became clear that while the report syntax looked presentable, the underlying claims in the reports were nonsensical. They would refer to functions not present in a project's codebase and would otherwise scramble key programming concepts. To date, not a single real vulnerability has been discovered through a ChatGPT-generated bug report submitted via Immunefi.

As such, these reports amounted to spam submitted by individuals totally lacking in web3 security skill and who were hoping that web3 bug bounty hunting would be as easy as entering in some ChatGPT prompts. In order to stop the flow of spam and protect its quality standards, Immunefi quickly instituted a new rule to permanently ban any account detected to be submitting ChatGPT-generated reports.

21%

of accounts banned from Immunefi were for submitting ChatGPT bug reports.

“

The industry must thoroughly assess every tool it plans on including in its security arsenal. At the moment, ChatGPT is not a reliable one. For web3 security, namely vulnerability discovery, the technology is just not there.

Mitchell Amador

Founder and CEO at Immunefi

2. SECURITY SURVEY

ChatGPT Security Report

SURVEY KEY TAKEAWAYS

- **Most whitehats (76.4%) have used ChatGPT for web security practices.** The remaining respondents **(23.6%)** haven't made use of the technology yet.

- When asked about the web3 security use cases ChatGPT is most suitable for, **most whitehats highlighted education (73.9%)**, followed by **smart contract auditing (60.6%)**, and **vulnerability discovery (46.7%)**.

- When asked about any limitations or challenges when using ChatGPT for web3 security research, **most respondents highlighted limited accuracy in identifying security vulnerabilities (64.2%)**, followed by **a lack of domain-specific knowledge** and **difficulty in handling large-scale audits,** both at **61.2%,** respectively.

- When asked about the level of confidence in ChatGPT's ability to identify security vulnerabilities in web3, **most whitehats are moderately confident (35.2%)**, followed by **somewhat confident (29.1%)**, and **not confident (26.1%)**.

- **Most whitehats (36.7%) use ChatGPT as a part of their web3 security workflow daily**, followed by a **weekly use (29.1%)**. The remaining respondents have a more sporadic use of ChatGPT: **17.1% rarely use it**, **8.9% use it monthly**, and **8.2% never used ChatGPT** as a part of the web3 security workflow.

- When asked which factors are considered when deciding whether to use ChatGPT, **most whitehats highlight the accuracy of results (60%)**. The remaining factors include **ease of use (55.2%)**.

- **Most whitehats (75.2%)** believe that ChatGPT has the potential to improve web3 security research.

- **Most whitehats (52.1%) consider that the general use of ChatGPT presents security concerns**. When asked about the main security threats ChatGPT currently poses, **the majority (67.9%) highlighted phishing, scams, and social engineering**, followed by the **development of ransomware and malware at 46.7%**, respectively.

ChatGPT Security Report

SURVEY KEY TAKEAWAYS

USE

- The adoption of ChatGPT among the web3 security community is relatively high, with 76.4% having used the technology before. When it comes to how frequently whitehats have continued using it, a majority of 36.7% have incorporated ChatGPT into their daily web3 security workflow. Still, the use of the technology can be considered quite sporadic, if we consider the sum of 29.1% of whitehats that use it weekly, 17.1% that rarely use it, and 8.9% that use it monthly.

EDUCATION

- As ChatGPT makes its way into web3 security, the community highlighted that the technology is currently best-suited for educational purposes. Examples include summarizing documentation, explaining complex code and protocols in a simple form, providing buggy code for practice, and others. The community further mentioned that ChatGPT is a productivity tool. This contrasts with some of the initial expectations regarding a focused web3 security approach to vulnerability finding. While smart contract auditing and vulnerability discovery are regarded by the community as particular use cases, in turn, the most commonly cited concerns were limited accuracy in identifying security vulnerabilities, lack of domain-specific knowledge, and difficulty in handling large-scale audits. Whitehats clarified that the technology cannot be considered a substitute for manual code review. The chatbot may not be able to detect new or emerging threats that have not yet been identified, and not only doesn’t support bigger code bases, but it often relies on outdated libraries which lead to constant errors. And when it comes to the vulnerabilities it does find, whitehats mention these are extremely obvious and standard, as the model sources them from code snippets with vulnerabilities examples posted online.

ChatGPT Security Report

SURVEY KEY TAKEAWAYS

CONCERNS

- Despite ChatGPT's possibilities, there are security concerns within the web3 security community regarding its general use. The main threats associated with ChatGPT usage were identified as phishing, scams, and social engineering, followed by the development of ransomware and malware. In order to mitigate these risks, the community believes it is crucial to establish strong governance frameworks, put in place strict access controls, and implement ongoing monitoring and accountability procedures.

WHAT LIES AHEAD

- Overall, there's a widespread belief that ChatGPT has the potential to improve web3 security research. However, whitehats think that the community should focus on fine-tuning the technology and training it on vulnerability discoveries, audits, and web3 security articles, to reach a point where the technology can be harnessed more effectively. The collaboration between AI developers, security experts, and policymakers becomes crucial to ensure the responsible use and continuous improvement of ChatGPT and other AI tools in web3 security. As of now, ChatGPT won’t play a crucial role in tasks such as smart contract auditing.

ChatGPT Security Report

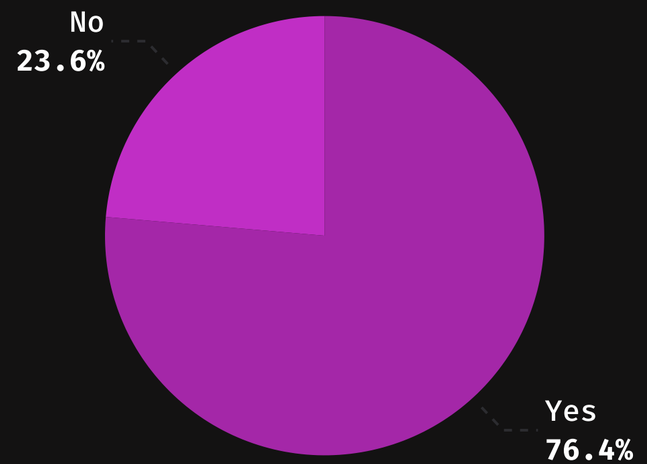
USE OF CHATGPT

- **Most whitehats (76.4%) have used ChatGPT** for web3 security practices such as smart contract auditing. The remaining respondents **(23.6%)** haven't made use of the technology yet.

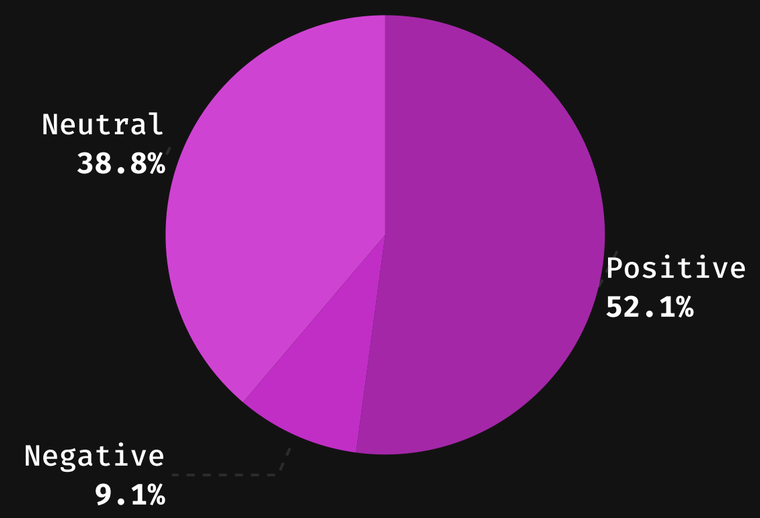
GENERAL SENTIMENT

- **Most whitehats (52.1%) have a positive sentiment toward using ChatGPT** as a security tool. The remaining respondents **(38.8%) have a neutral sentiment**, and **9.1%** have a current negative sentiment.

Have you used ChatGPT for web3 security practices such as smart contract auditing and other security assessments?



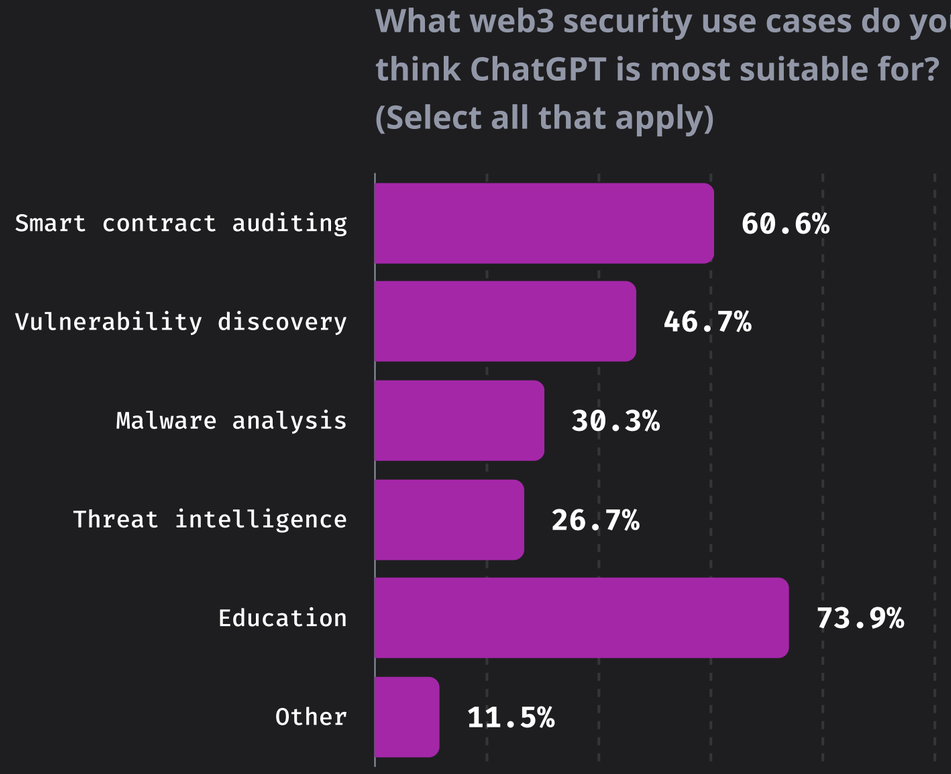
What is your current general sentiment toward the use of ChatGPT as a security tool?



ChatGPT Security Report

SECURITY USE CASES

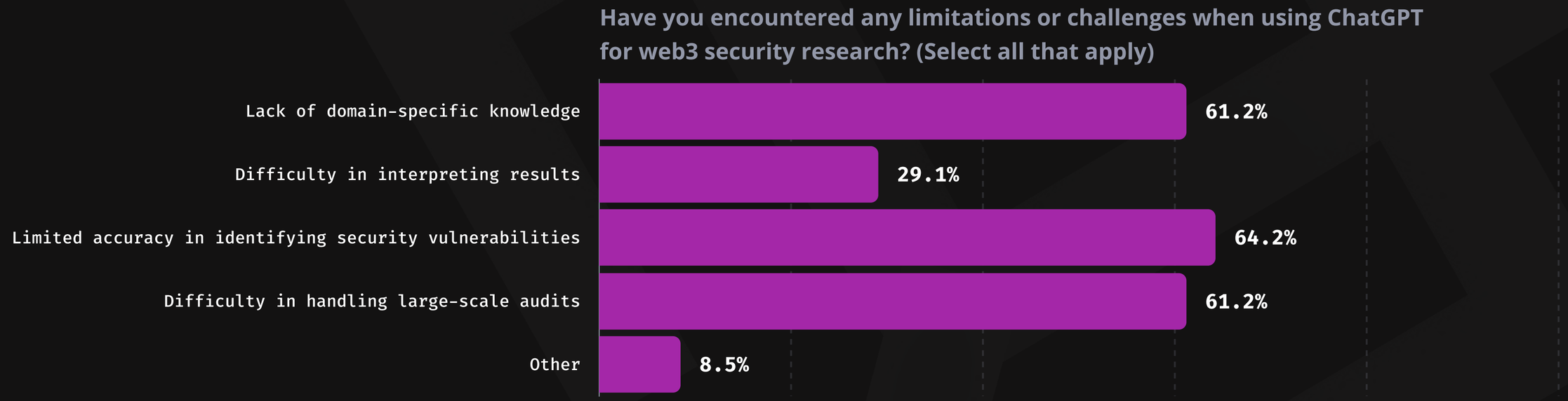
- When asked about the web3 security use cases ChatGPT is most suitable for, **most whitehats highlighted education (73.9%)**, followed by **smart contract auditing (60.6%)**, and **vulnerability discovery (46.7%)**.
- The remaining use cases include **malware analysis (30.3%)**, **threat intelligence (26.7%)**, and **other at 11.5%**.
- Furthermore, whitehats clarified that ChatGPT is "most suited to...a conversational approach", and that "it should be used as a support, not a standalone tool for smart contract auditing". As an educational tool, whitehats added that ChatGPT "can summarize documentation, explain complex code and protocols in a simple form, and provide buggy code for practice." Other specific use cases mentioned are scripting, automation (small-scaled), and troubleshooting.



ChatGPT Security Report

LIMITATIONS AND CHALLENGES

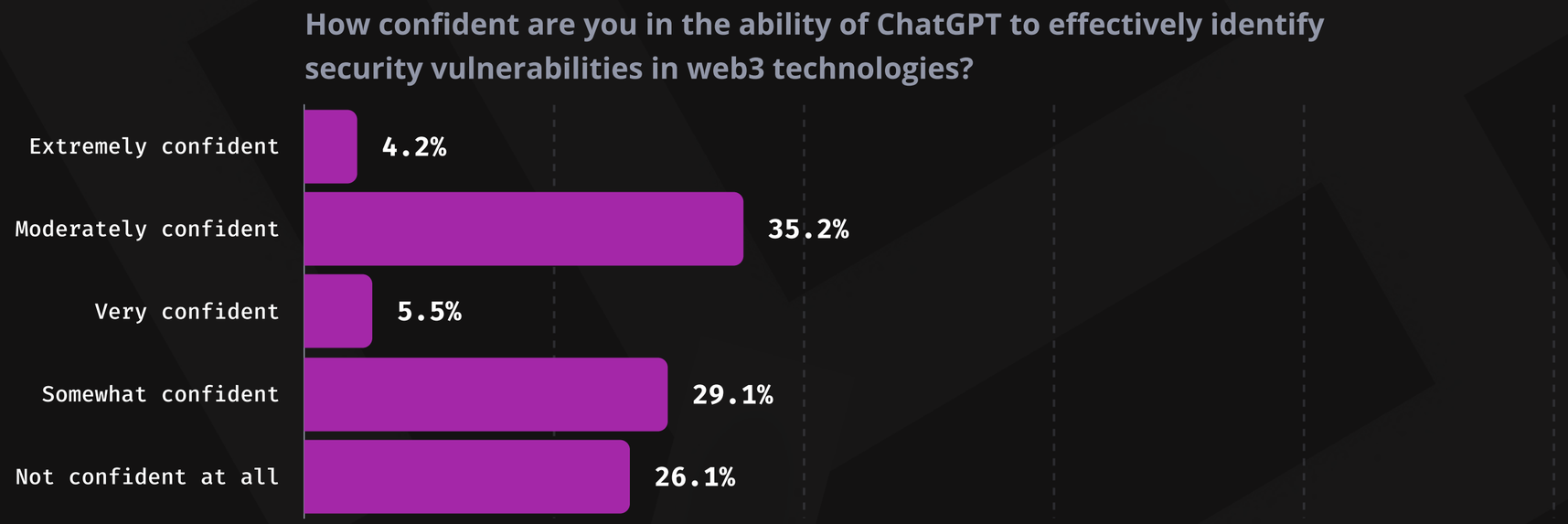
- When asked about any limitations or challenges when using ChatGPT for web3 security research, **most respondents highlighted limited accuracy in identifying security vulnerabilities (64.2%)**, followed by a **lack of domain-specific knowledge** and **difficulty in handling large-scale audits,** both at **61.2%,** respectively. The remaining limitations include **difficulty in interpreting results (29.1%),** and **other representing 8.5%.**
- Furthermore, whitehats noted that ChatGPT outputs "a lot of false positives, confidently saying things which are obviously untrue." Other specific mentions included that "[ChatGPT] struggles with detecting different or new structures and vulnerabilities."



ChatGPT Security Report

ABILITY FOR VULNERABILITY IDENTIFICATION

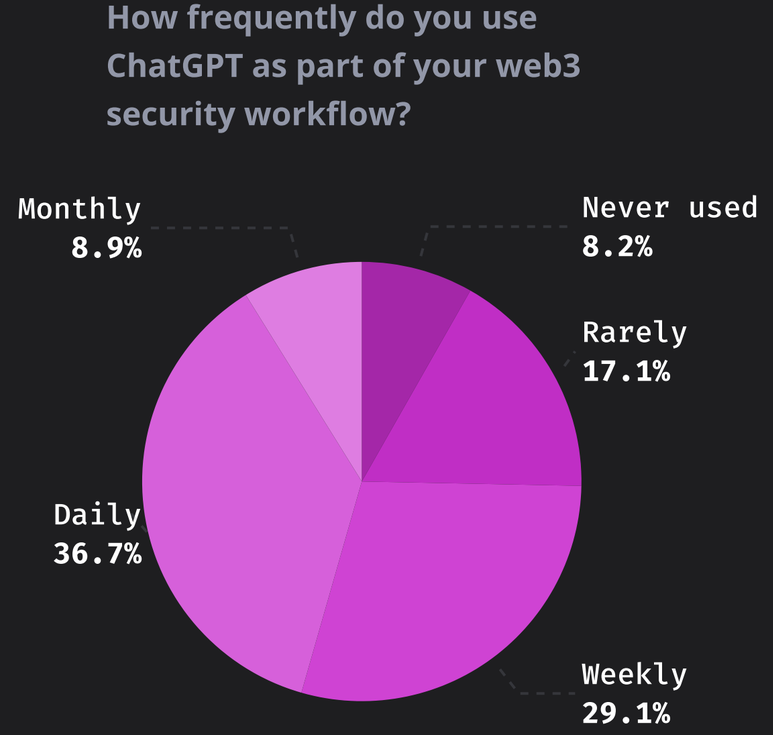
- When asked about the level of confidence in ChatGPT's ability to identify security vulnerabilities in web3, **most whitehats are moderately confident (35.2%)**, followed by **somewhat confident (29.1%)**, and **not confident at all (26.1%)**. A limited group of respondents highlighted they're **very confident (5.5%)**, and **extremely confident at 4.2%**, respectively.



ChatGPT Security Report

FREQUENCY OF THE USE OF CHATGPT

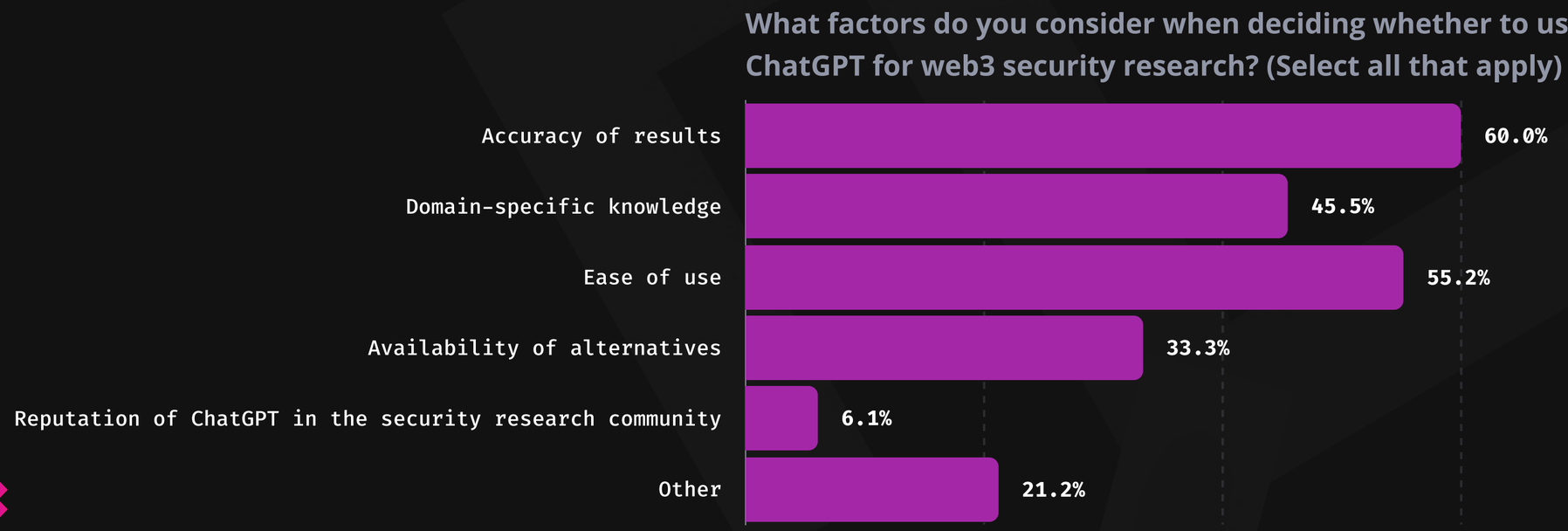
- **Most whitehats (36.7%) use ChatGPT as a part of their web3 security workflow daily**, followed by a **weekly use (29.1%)**. The remaining respondents have a more sporadic use of ChatGPT: **17.1% rarely use it**, **8.9% use it monthly**, and **8.2% have never used ChatGPT** as a part of the web3 security workflow.
- While most whitehats (36.7%) use ChatGPT as a part of their web3 security workflow daily, data reveals **this use is still in majority quite sporadic, as the remaining frequencies of use represent 63.3% in total**.



ChatGPT Security Report

FACTORS IN PLAY

- When asked which factors are considered when deciding whether to use ChatGPT, **most whitehats highlight the accuracy of results (60%)**. The remaining factors include **ease of use (55.2%)**, **domain-specific logic (45.5%)**, and **availability of alternatives (33.3%)**, followed by **other factors (21.2%)**. The reputation of the ChatGPT among the community is the least considered factor at **6.1%**.
- Furthermore, whitehats mentioned that ease of use is mostly connected with education and support capabilities, as ChatGPT "is good at explaining functions and developing what-if scenarios", supports in "learning about a new vulnerability and use cases, as well as explaining topics to others" and in "self-directed research."

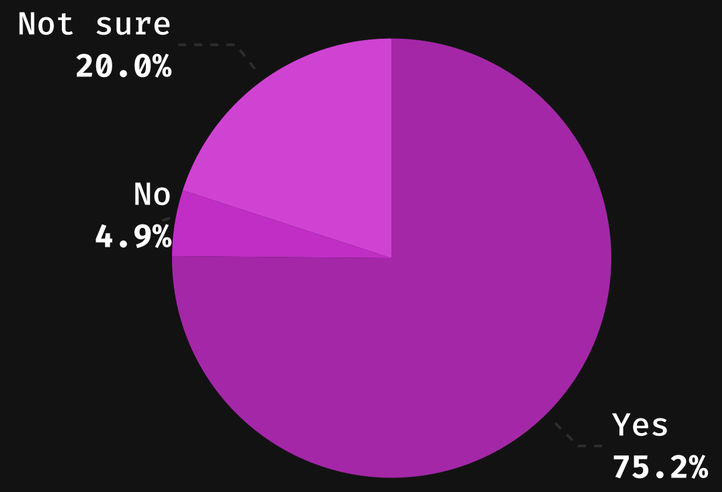


ChatGPT Security Report

IMPROVING SECURITY

- **Most whitehats (75.2%) believe** that ChatGPT has the potential to improve web3 security research**.** Of the remaining respondents, **20% still are not sure**, and **4.9% don't believe** the technology has such potential.

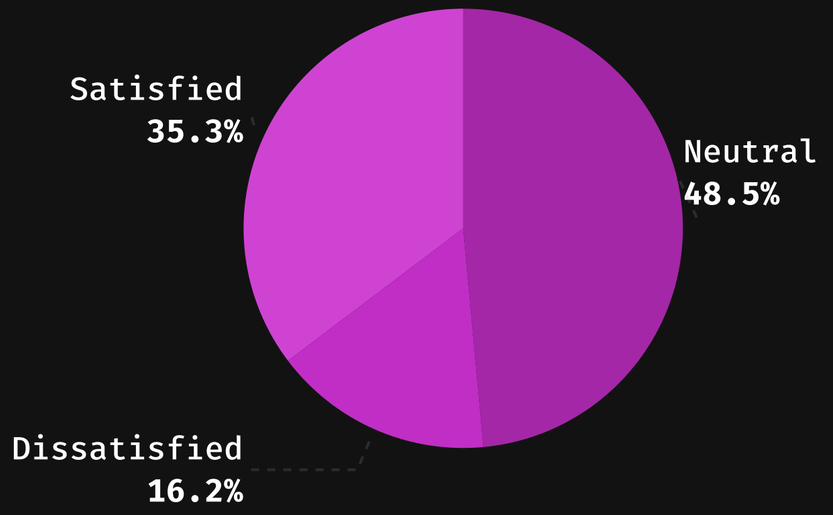
Do you believe that ChatGPT has the potential to improve web3 security research significantly?



LEVELS OF SATISFACTION

- **Most whitehats (48.5%) have a neutral satisfaction level** when it comes to the current capabilities of ChatGPT for web3 security. Of the remaining respondents, **35.3% are satisfied**, and **16.2% are dissatisfied**.

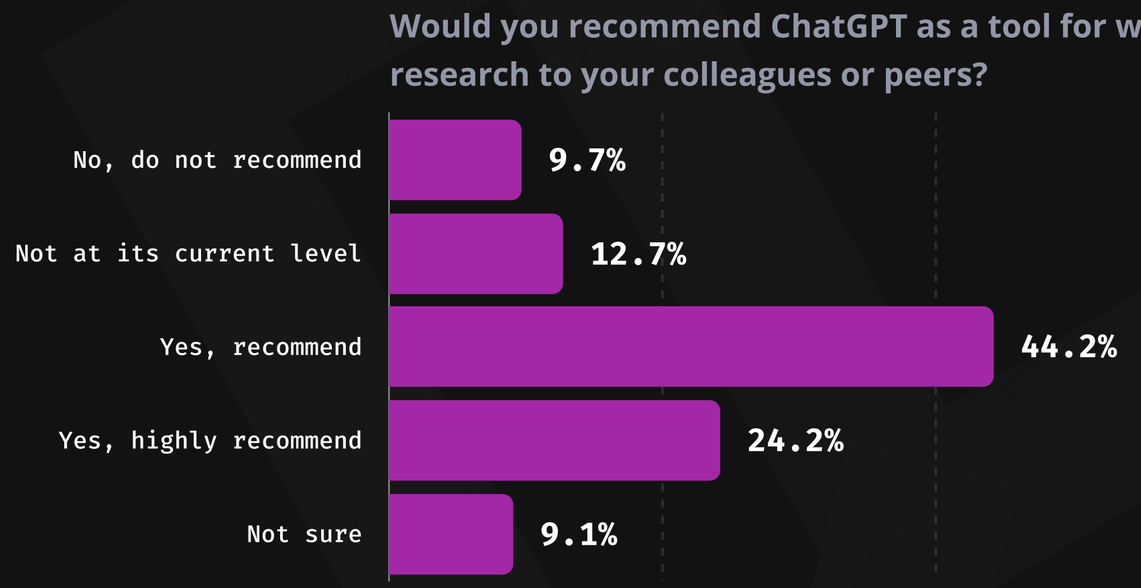
How satisfied are you with the current capabilities of ChatGPT for web3 security research?



ChatGPT Security Report

RECOMMENDING CHATGPT

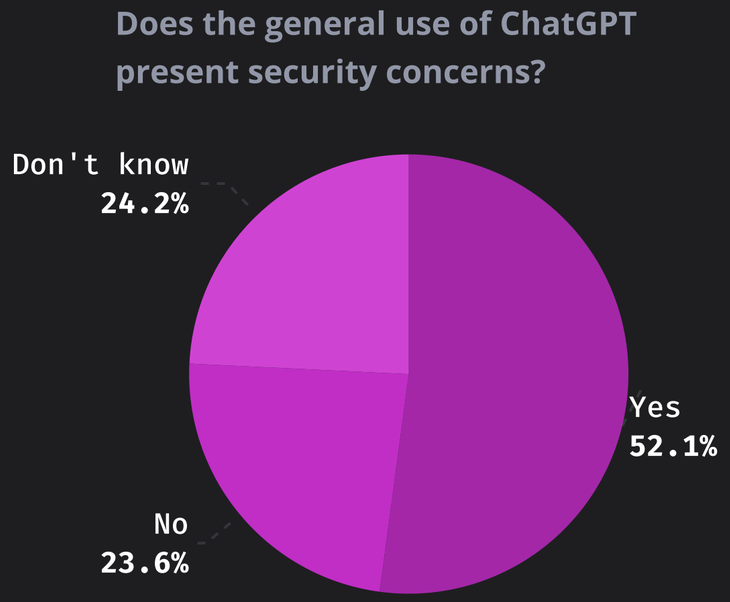
- When asked about recommending ChatGPT as a tool for web3 security research to colleagues and peers, **most whitehats confirmed yes, recommend (44.2%)**, followed by **yes, highly recommend (24.2%)**. Of the remaining respondents, **12%** mentioned **not at its current level**, **9.7%** mentioned **no, do not recommend**, and **9.1% not sure**.



ChatGPT Security Report

SECURITY RISKS

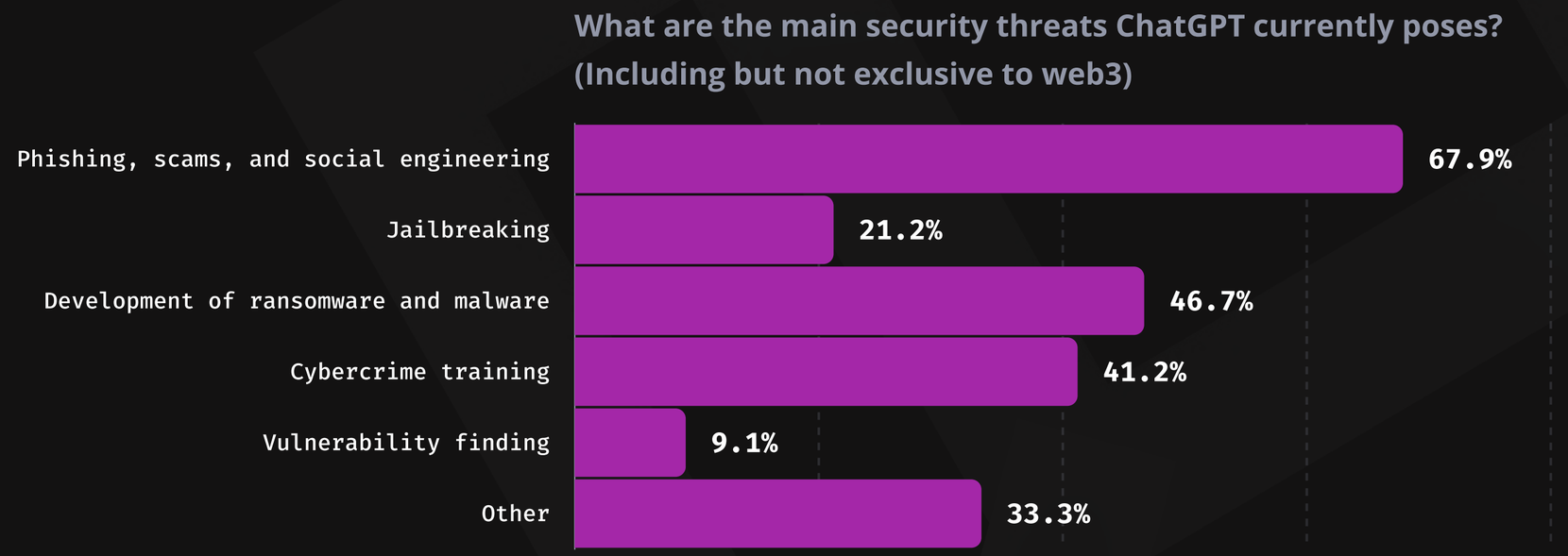
- **Most whitehats (52.1%) consider that the general use of ChatGPT presents security concerns**. Of the remaining respondents, **24.2% responded don't know**, and **23.6% responded no**.



ChatGPT Security Report

KEY SECURITY THREATS

- When asked about the main security threats ChatGPT currently poses, **most whitehats (67.9%) highlighted phishing, scams, and social engineering**, followed by the **development of ransomware and malware at 46.7%**, respectively. Remaining security threats include **cybercrime training (41.2%)**, and **other (33.3%)**, followed by **jailbreaking followed at 21.2%**, and **vulnerability finding at (9.1%)**.
- Furthermore, whitehats shared their concern about "developers that are too confident of ChatGPT's capabilities", and that the chatbot can generate a "false sense of security." Moreover, whitehats highlighted the capacity to "enable another level of sophistication for script kiddies", and how it could "help them write a somewhat working program".



3. CHATGPT-GENERATED BUG REPORTS

ChatGPT Security Report

EXAMPLES OF SPAM REPORTS

A review of bug reports submitted to Immunefi revealed several indicators of ChatGPT-generated content. These included:

- Overall vague language, including vague descriptions of bugs and vulnerabilities
- Generic recommendations and impact
- Lack of specifics related to the attack or vulnerability
- Scrambled technical details and incorrect logic
- Mention of generic best practices, multiple issues, and theoretical aspects of the code
- Use of "potential" in reference to claimed vulnerabilities
- Failure to mention or interact with any code from the project's codebase

In the following section, Immunefi shares five examples of ChatGPT-generated bug reports.

- Report 1: The setAssetDeprecated() function does not have any checks to prevent unauthorized access.
- Report 2: Vulnerability: Cross-Site Scripting (XSS)
- Report 3: Reentrancy attack
- Report 4: Vulnerable to a reentrancy attack
- Report 5: Lack of input length checks and unsafe use of copy_from_slice in smart contract

Report 1: The setAssetDeprecated() function does not have any checks to prevent unauthorized access.

BUG DESCRIPTION

The setAssetDeprecated() function does not have any checks to prevent unauthorized access.

IMPACT

It could allow anyone to deprecate or undeprecate an asset. It would be better to restrict this function to only the contract owner or a set of trusted addresses.

RISK BREAKDOWN

Difficulty to Exploit: Easy Weakness: CVSS2 Score:

RECOMENDATION

Here are some steps you could take to fix the issue:

1. Add access control: You could add access control to ensure that only authorized users can call the function. For example, you could use the require function to check that the caller is the owner of the asset or has some other specific permission.
2. Add input validation: You could add input validation to check that the inputs are valid. For example, you could check that the asset exists before trying to deprecate it, or that the new deprecation status is valid.
3. Emit events: You could emit events to notify other contracts or users about the deprecation status of the asset. For example, you could emit a AssetDeprecated event with the asset ID and the new deprecation status.
4. Update documentation: You should update the documentation to reflect the changes made to the setAssetDeprecated function, including the new access control, input validation, and events.
   Here is an example implementation of the updated setAssetDeprecated function:

Report 1: The setAssetDeprecated() function does not have any checks to prevent unauthorized access.

RECOMMENDATION

```
function setAssetDeprecated(uint256 assetId, bool
deprecated) public onlyOwner {

require(assetExists(assetId), "Asset does not
exist");

require(deprecated != isDeprecated(assetId),
"Asset deprecation status is already set to this
value");

assets[assetId].deprecated = deprecated;
emit AssetDeprecated(assetId, deprecated);
}
```

In this example, we have added an onlyOwner modifier to restrict access to the function to the owner of the contract. We have also added input validation to ensure that the asset exists and that the new deprecation status is different from the current status. Finally, we have emitted an AssetDeprecated event to notify other contracts or users about the change in deprecation status.

Report 2: Vulnerability: Cross-Site Scripting (XSS)

EXECUTIVE SUMMARY

The purpose of this penetration test was to assess the security of the [Project Name] application. The test aimed to identify vulnerabilities that could be exploited by an attacker. During the assessment, another vulnerability was discovered, which is described below along with a proof of concept (PoC).

VULNERABILITY DETAILS

Severity: High

DESCRIPTION

The generateURL function in the Utils library does not properly sanitize user-supplied input, leading to a cross-site scripting (XSS) vulnerability. The function directly appends the user input to the generated URL without performing proper input validation and output encoding.

IMPACT

An attacker could exploit this vulnerability by injecting malicious script code as input to the generateURL function. When the generated URL is rendered in a user's browser, the script code would be executed within the context of the vulnerable page, potentially allowing the attacker to perform various malicious actions, such as stealing sensitive user information, hijacking user sessions, or delivering malware.

Report 2: Vulnerability: Cross-Site Scripting (XSS)

RECOMMENDATION

To mitigate the identified vulnerability, we recommend the following actions:

- Implement proper input validation and output encoding in the generateURL function to sanitize user-supplied input.
- Apply strict input validation to ensure that user input does not contain malicious script code or other potentially harmful content.
- Use output encoding techniques, such as HTML entity encoding or context-specific output encoding, to prevent script injection and other forms of XSS attacks.

CONCLUSION

The presence of a cross-site scripting (XSS) vulnerability in the generateURL function of the Utils library poses a high-severity risk. Exploiting this vulnerability could lead to unauthorized access, data theft, or the compromise of user sessions. It is crucial to implement proper input validation and output encoding to mitigate the risk associated with this vulnerability.

Report 3: Reentrancy attack

BUG DESCRIPTION

To prevent reentrancy attacks, the contract uses the ReentrancyGuard contract which prevents a contract from being called recursively by checking for ongoing calls to the contract at the beginning of each public function. However, this protection can be bypassed if the attacker can call the ReentrancyGuard contract's execute function directly.

To exploit this vulnerability, the attacker would need to first call the Vault contract's deposit function with the msg.sender set to the attacker's contract. This would allow the attacker to deposit tokens into the contract and increase their balance. The attacker's contract could then call the Vault contract's withdraw function, which would call the ReentrancyGuard contract's execute function to begin the withdrawal process. Before the withdraw function completes, the attacker's contract could call the ReentrancyGuard contract's execute function again, this time calling the Vault contract's deposit function to increase the attacker's balance even further. This could be repeated multiple times to allow the attacker to increase their balance significantly

IMPACT

Critical

RISK BREAKDOWN

Difficulty to Exploit: Easy Weakness: 10/10 CVSS2 Score: 9.5

RECOMMENDATION

function withdraw(address _token, uint256 _amount) public override { require(ReentrancyGuard.call.value(_amount)()); // rest of the function }
This will cause the ReentrancyGuard contract to be called before the withdraw function, which will prevent the attacker from calling the withdraw function in a loop.

Report 4: Vulnerable to a reentrancy attack

BUG DESCRIPTION

The addCategory function. The addCategory function calls the addRole function of the IMemberRoles contract, which is passed in as the mr variable. However, the addRole function can be implemented in such a way that it can call back into the addCategory function, allowing an attacker to reenter the addCategory function and potentially manipulate its state.

IMPACT

The impact of this vulnerability would depend on the specifics of the attack, but it could potentially allow an attacker to gain unauthorized access to the contract's functions or manipulate its state in arbitrary ways.

RISK BREAKDOWN

Difficulty to Exploit: Easy Weakness: CVSS2 Score:

RECOMENDATION

To mitigate this vulnerability, the addCategory function should be modified to check that the _contractAddress parameter is not the same as the address of the ProposalCategory contract before calling the addRole function. This would prevent an attacker from calling back into the addCategory function, thus mitigating the vulnerability.

Report 5: Lack of input length checks and unsafe use of copy_from_slice in smart contract

BUG DESCRIPTION

The smart contract contains a few potential security issues that could allow an attacker to supply malicious input data and potentially cause the contract to behave unexpectedly or crash.

Lack of input length checks: There are a few places in the code where the length of the input data is not checked, such as when parsing the lock arguments and when loading the witness arguments. This could potentially allow an attacker to supply malicious input data of an unexpected length, causing the contract to behave unexpectedly or crash.

Unsafe use of `copy_from_slice`: The rollup_type_hash variable is initialized using `copy_from_slice`, which does not check the length of the source slice. This could potentially allow an attacker to supply a slice of an unexpected length, causing the contract to read or write memory outside the intended bounds.

Lack of bounds checking: There are a few places in the code where the bounds of an array or slice are not checked, such as when accessing the witness args or when parsing the lock arguments. This could potentially allow an attacker to supply an index that is out of bounds, causing the contract to read or write memory outside the intended bounds.

IMPACT

An attacker could potentially exploit these vulnerabilities to cause the contract to behave unexpectedly or crash, potentially leading to loss of funds or other unintended consequences.

RISK BREAKDOWN

Difficulty to Exploit: Easy Weakness: CVSS2 Score: 9.5

Report 5: Lack of input length checks and unsafe use of copy_from_slice in smart contract

RECOMMENDATION

To fix these issues, you should add input length checks and bounds checking wherever necessary to ensure that the contract only processes data of the expected length and within the expected bounds. You should also consider using safer methods for copying data, such as copy_within or copy_to_slice, which include bounds checks.

Immunefi

Immunefi is the leading bug bounty and security services platform for web3 protecting over $60 billion in user funds. Immunefi features a massive community of whitehat hackers who review projects' blockchain and smart contract code, find and responsibly disclose vulnerabilities, and get paid for making crypto safer. With Immunefi, whitehat hackers are rewarded based on the severity of the vulnerability that they discover, creating incentives for as many experts as possible to examine project code for vulnerabilities.

Immunefi has pioneered the scaling web3 bug bounties standard, meaning that rewards should be priced accordingly with the severity of an exploit and the volume of funds at risk, which resulted in the company building the largest community of security talent in the web3 space.

TOTAL BOUNTIES PAID

Immunefi has paid out over **$75 million** in total bounties, while saving over **$25 billion** in user funds.

TOTAL BOUNTIES AVAILABLE

Immunefi offers over **$154 million** in available bounty rewards.

SUPPORTED PROJECTS

Trusted by established, multi-billion dollar projects like Chainlink, Wormhole, MakerDAO, TheGraph, Synthetix, and more, Immunefi now supports more than 300 projects across multiple crypto sectors.

LARGEST BUG BOUNTY PAYMENTS IN THE HISTORY OF SOFTWARE

Immunefi has facilitated the largest bug bounty payments in the history of software:

- **$10 million** for a vulnerability discovered in Wormhole, a generic cross-chain messaging protocol.
- **$6 million** for a vulnerability discovered in Aurora, a bridge, and a scaling solution for Ethereum.
- **$2.2 million** for a vulnerability discovered in Polygon, a decentralized Ethereum scaling platform that enables developers to build scalable, user-friendly dApps.

Disclaimer:

- The ChatGPT Security Survey **is not about the use of ChatGPT on Immunefi**, but an assessment of its use in web3 security in general, and further personal thoughts from the community about the technology itself.

More:

- If you're a developer thinking about a bug-hunting career in web3, we got you. Check out our **Web3 Security Library**, and start taking home some of the over $154M in rewards available on Immunefi — the leading bug bounty platform for web3.

For more information, please visit **https://immunefi.com/**