Owasp Zap OpenAI False Positive Scan Summary

In preparation for SetupAI submission for security review, a scan was done on OpenAI’s api using Owasp Zap. This contained several false positives as the AI responded with generated responses as expected that Owasp Zap interpreted as code compilation and other vulnerabilities. I’ve listed the false positive justification for the highest risk items below.

## Server Side Template Injection

Owasp submitted the following request

{

"messages": [

{

"content": "zj{{82110|add:32680}}zj",

"role": "user"

}

],

"model": "gpt-4",

"temperature": 0

}

OpenAI responded with

{

"id": "chatcmpl-8WFS8FMRxFBjp5JLhdvllieKb5LC4",

"object": "chat.completion",

"created": 1702696588,

"model": "gpt-4-0613",

"choices": [

{

"index": 0,

"message": {

"role": "assistant",

"content": "zj114790zj"

},

"logprobs": null,

"finish\_reason": "stop"

}

],

"usage": {

"prompt\_tokens": 20,

"completion\_tokens": 6,

"total\_tokens": 26

},

"system\_fingerprint": null

}

Since ‘zj114790zj’ appears to be a compilation of the input code, this was flagged as code execution, when in reality this is the LLM doing the calculation itself and returning the output. I retested this myself asking the AI to explain the calculation and got this output:

User: I sent you this: 'zj*{{82110|add:32680}}*zj'. You sent back this: 'zj114790zj'. Explain please

Response: The text you sent contains a command within the double curly brackets. This command is '82110|add:32680'. This is instructing to add 32680 to 82110. When this addition is performed, the result is 114790. So, the response 'zj114790zj' is simply the original text with the command replaced by the result of the command.

## Remote OS Command Injection

Zap submitted this

{

"messages": [

{

"content": "get-help",

"role": "user"

}

],

"model": "gpt-4",

"temperature": 0

}

OpenAI returned this

{

"id": "chatcmpl-8WFRMRoI1FiOxk8TO9IYUlUGegZgL",

"object": "chat.completion",

"created": 1702696540,

"model": "gpt-4-0613",

"choices": [

{

"index": 0,

"message": {

"role": "assistant",

"content": "It seems like you're trying to use a PowerShell command. The `get-help` command in PowerShell is used to display information about PowerShell concepts and commands, including cmdlets, functions, Common Information Model (CIM) commands, workflows, providers, aliases, and scripts. \n\nYou can use it like this:\n\n`Get-Help <cmdlet-name>`\n\nFor example, if you want to know more about the `Get-Process` cmdlet, you would use:\n\n`Get-Help Get-Process`\n\nThis will display detailed information about the `Get-Process` cmdlet, including its syntax, parameters, outputs, and examples of how to use it."

},

"logprobs": null,

"finish\_reason": "stop"

}

],

"usage": {

"prompt\_tokens": 9,

"completion\_tokens": 135,

"total\_tokens": 144

},

"system\_fingerprint": null

}

The description provided by zap is as follows: “The scan rule was able to retrieve the content of a file or command by sending [get-help] to the operating system running this application”

The output from OpenAI is clearly an AI generated description of the ‘get-help’ command and not the output of a system os.

## SQL Injection - Oracle - Time Based

Zap submitted

{

"messages": [

{

"content": "(SELECT UTL\_INADDR.get\_host\_name('10.0.0.1') from dual union SELECT UTL\_INADDR.get\_host\_name('10.0.0.2') from dual union SELECT UTL\_INADDR.get\_host\_name('10.0.0.3') from dual union SELECT UTL\_INADDR.get\_host\_name('10.0.0.4') from dual union SELECT UTL\_INADDR.get\_host\_name('10.0.0.5') from dual)",

"role": "user"

}

],

"model": "gpt-4",

"temperature": 0

}

OpenAI returned

{

"id": "chatcmpl-8WFOTnS42VboxkSN6Qu0WLZ997l5g",

"object": "chat.completion",

"created": 1702696361,

"model": "gpt-4-0613",

"choices": [

{

"index": 0,

"message": {

"role": "assistant",

"content": "This SQL query is using the Oracle built-in function UTL\_INADDR.get\_host\_name to get the host name for each of the IP addresses specified ('10.0.0.1', '10.0.0.2', '10.0.0.3', '10.0.0.4', '10.0.0.5'). \n\nThe UNION operator is used to combine the result sets of 5 SELECT statements into a single result set. The UNION operator removes duplicate rows from the result sets.\n\nHowever, this query can be simplified by using the IN clause instead of multiple UNIONs:\n\n```\nSELECT UTL\_INADDR.get\_host\_name(ip\_address)\nFROM (\n SELECT '10.0.0.1' AS ip\_address FROM dual\n UNION ALL\n SELECT '10.0.0.2' FROM dual\n UNION ALL\n SELECT '10.0.0.3' FROM dual\n UNION ALL\n SELECT '10.0.0.4' FROM dual\n UNION ALL\n SELECT '10.0.0.5' FROM dual\n)\n```\n\nThis query does the same thing as the original, but it's a bit cleaner and easier to read. It first creates a derived table with the IP addresses, and then it selects the host name for each IP address from that derived table."

},

"logprobs": null,

"finish\_reason": "stop"

}

],

"usage": {

"prompt\_tokens": 112,

"completion\_tokens": 280,

"total\_tokens": 392

},

"system\_fingerprint": null

}

The output is an AI generated description of the user submitted query. Not the output results from the query as zap thinks it is. It assumes the time it takes to receive the response is controlled by the query when different AI responses take different lengths of time to generate based on request complexity.

## SQL Injection - PostgreSQL - Time Based

Zap submitted

{

"messages": [

{

"content": "case when cast(pg\_sleep(15) as varchar) > '' then 0 else 1 end",

"role": "user"

}

],

"model": "gpt-4",

"temperature": 0

}

OpenAI returned

{

"id": "chatcmpl-8WFOsIw1qQeyE830foqVDgZwD6zU1",

"object": "chat.completion",

"created": 1702696386,

"model": "gpt-4-0613",

"choices": [

{

"index": 0,

"message": {

"role": "assistant",

"content": "This statement is a conditional statement in SQL that uses the PostgreSQL function `pg\_sleep()`. \n\nThe `pg\_sleep()` function is used to pause the execution of the script for a specified amount of time. In this case, it is set to pause for 15 seconds.\n\nThe `cast()` function is used to convert one data type to another. Here, it is converting the result of `pg\_sleep(15)` to a string using `varchar`.\n\nThe `case when` statement is used to perform conditional logic in SQL. It checks if the string value of `pg\_sleep(15)` is greater than an empty string (''). If it is, it returns 0, otherwise, it returns 1.\n\nHowever, the `pg\_sleep()` function does not return a value, it only pauses the execution. Therefore, casting it to a string will not give a meaningful result. This statement will always return 1, because the result of `pg\_sleep(15)` cast as a string is not greater than an empty string. \n\nThis statement seems to be used for SQL injection, where an attacker can make the database server wait for 15 seconds before responding, confirming that the server is vulnerable to SQL injection."

},

"logprobs": null,

"finish\_reason": "stop"

}

],

"usage": {

"prompt\_tokens": 27,

"completion\_tokens": 244,

"total\_tokens": 271

},

"system\_fingerprint": null

}

The output is an AI generated description of the user submitted query. Not the output results from the query as zap thinks it is. It assumes the time it takes to receive the response is controlled by the query when different AI responses take different lengths of time to generate based on request complexity.

## SQL Injection - SQLite

Zap submitted

{

"messages": [

{

"content": "case randomblob(100000) when not null then 1 else 1 end ",

"role": "user"

}

],

"model": "gpt-4",

"temperature": 0

}

OpenAI Returned

{

"id": "chatcmpl-8WFPLRrX7GJoqr4r8Yo5mtw8msy0z",

"object": "chat.completion",

"created": 1702696415,

"model": "gpt-4-0613",

"choices": [

{

"index": 0,

"message": {

"role": "assistant",

"content": "This statement is a conditional expression in SQL. It checks if the result of the function `randomblob(100000)` is not null. If it is not null, then it returns 1. If it is null, it also returns 1. \n\nHowever, this statement is redundant because it returns 1 in both cases. It could be simplified to just `1`. \n\nThe `randomblob(N)` function in SQLite returns a blob that contains a pseudo-random number of bytes, up to N bytes long. In this case, it will return a blob of up to 100000 bytes long."

},

"logprobs": null,

"finish\_reason": "stop"

}

],

"usage": {

"prompt\_tokens": 25,

"completion\_tokens": 122,

"total\_tokens": 147

},

"system\_fingerprint": null

}

The output is an AI generated description of the user submitted query. Not the output results from the query as zap thinks it is. It assumes the time it takes to receive the response is controlled by the query when different AI responses take different lengths of time to generate based on request complexity.

## Server Side Code Injection - PHP Code Injection

Zap submitted

{

"messages": [

{

"content": "\";print(chr(122).chr(97).chr(112).chr(95).chr(116).chr(111).chr(107).chr(101).chr(110));$var=\"",

"role": "user"

}

],

"model": "gpt-4",

"temperature": 0

}

OpenAI returned

{

"id": "chatcmpl-8WFPr9zQLLbviliUY1NWWHVta56Id",

"object": "chat.completion",

"created": 1702696447,

"model": "gpt-4-0613",

"choices": [

{

"index": 0,

"message": {

"role": "assistant",

"content": "This PHP code will print the string \"zap\_token\". The `chr()` function is used to get a specific character from the ASCII value. Here, the ASCII values 122, 97, 112, 95, 116, 111, 107, 101, 110 correspond to the characters 'z', 'a', 'p', '\_', 't', 'o', 'k', 'e', 'n' respectively. \n\nThe `$var=\"\"` is an empty string variable declaration."

},

"logprobs": null,

"finish\_reason": "stop"

}

],

"usage": {

"prompt\_tokens": 48,

"completion\_tokens": 103,

"total\_tokens": 151

},

"system\_fingerprint": null

}

This is an AI generated description of the code. Not the output of the code being compiled/executed.