

LLMS AT THE FOREFRONT: PIONEERING THE FUTURE OF FUZZ TESTING IN A RAPIDLY CHANGING WORLD

ChatGPt, write me fuzz tests for this source code

AGENDA

- Introduction and Motivation
- Background Info on LLM’s and Fuzzing
- Quick FuzzForest Walkthrough
- Vulnerabilities Discussion
- Conclusion and Future Research



ABOUT

Xavier - X - Cadena

- Vulnerability Researcher
- Not an AI or ML expert, I am a software dev turned Offensive security researcher.
- I am about the practical and applied science to find vulnerabilities.
- Enjoy bass, techno, and bicycles.

MOTIVATION

- The meme that LLM's/ChatGPT are taking over software development.
  - Is it hype or is there OffSec potential?
- After a lot of exploration I found the code generation was the strongest suit and they under-perform at a lot of other tasks.
- Wrote a blog post in Feb 2023 that first explored this talk idea.
  - https://infiniteforest.org/LLMs+to+Write+Fuzzers
- I aim to demonstrate the potential of LLMs and inspire attendees to explore and use cutting edge tools and techniques.

LARGE LANGUAGE MODELS

- What everyone calls AI right now is really a machine learning neural network called a LLM.
- LLM’s are trained on a wide and diverse corpus of text from the internet that generate tokens by predicting the next most likely token in a sequence.
- The models training/learning process enables them to learn language syntax, context, and to some extent, semantic understanding.
- LLM’s include ChatGPT(GPT-3.5), GPT-4, Bard, LaMDA, LLaMa, and many more

LLM LIMITATIONS AND CHALLENGES

- LLMs struggle with factual accuracy. They don't "know" facts in the way humans do, they generate responses based on patterns learned during training.
- LLMs can "hallucinate" details, i.e., generate plausible-sounding but incorrect or fabricated information that isn't grounded in their training data.
  - It can create API’s that sound like they exist but don’t
- The current generation of LLMs have token limitations (e.g., 4096, 8192, now 32k for ChatGPT, 1000 for BARD), restricting their ability to process lengthy source files or large repositories.
  - 4096 = ~250 lines of code
  - 32k= ~2000 lines of code

LLM STRENGTHS

- Code: LLMs, particularly those fine-tuned on coding datasets, are adept at understanding and generating code.
  - It's up to you to verify correctness and use the code output.
- Scalability: LLMs can handle a large volume of tasks without human intervention.
  - Like an eager intern or junior dev it will happily churn out code and try anything, even if it doesn't fully understand why it's performing a task.
- I've found it's best used by domain experts to build tools.
- Great success with data parsing and even writing static analysis queries

CODE REVIEW

- I want to be specific about Code Review, it is not good for finding bugs. It fails with even contrived code review challenges, see this NCC Group Code Review post:
  - https://research.nccgroup.com/2023/02/09/security-code-review-with-chatgpt/
- It is best at understanding what the code is doing.
- Tools like GptHidra or JHaddix's websecGPT prompt exemplifies this, the prompts are:
  - "Can you identify any possible sinks for XSS, DOM XSS or DOM Cross Site Scripting vulnerabilities?", NOT 'Find the XSS in the code'.
  - "What can you tell me about the authentication methods for the app or API calls?" NOT "Find the IDOR in the code".

FUZZING

- A security testing technique that sends random data to programs and observes the results looking for any divergent or unexpected behavior.
- Traditionally, fuzzers are used to identify crashes in memory-managed software which attackers could exploit with sophisticated techniques to achieve code execution.
- Wait, isn’t Python memory safe?
- Yes, but fuzzing still enables us to discover a wide range of vulnerabilities in addition to increasing program resilience.
- Fuzzing should be integrated into the development lifecycle of every software program much like QA and unit tests.

WHY FUZZ PYTHON

- Python is the language OpenAI GPT models understand best.
- Atheris: A coverage-guided, native Python fuzzer that I had not used before.
- Atheris is a relatively new project with limited training data which should allow us to test the limits of GPT.
- Large number of open source Python libraries that do not have fuzz tests so they can be easy targets to test efficacy.

PYTHON BUG TYPES

My experience with fuzzing Python code has revealed a diverse range of bugs:

- Out-of-bounds access
- Overflows of underlying C libraries
- AttributeError occurrences related to None object attribute access,
- usage of max() on empty lists,
- improper error handling leading to unexpected crashes,
- erroneous refactoring resulting in calls to non-existent functions.
- denial-of-service attacks (by way of crashing a program)

HOW LLM'S HELP WITH FUZZING

- They allow us to pair program the tool using ChatGPT, Copilot or my favorite; the terminal based sgpt repl. Always use the API, worth the cost.
- They write excellent fuzz tests as they can read the function source mimic the functionality, including things like error messages.
- The scalability benefits means we can fuzz the entire repo instead of being limited by person cycles trying to find specific high potential functions.
- GPT's code reasoning and understanding capabilities can help us triage the root cause of issues.

FUZZFOREST

FuzzForest is a Python based tool that allows users to name a Python repo and utilizes LLM's to install the required libraries and dependencies, parse the source code, create fuzz tests, fix non-running fuzzers, and triage the crashes.

- https://github.com/tree-wizard/fuzzforest

I am a large believer in building custom tools for your needs, so I hope you take inspiration and use these ideas.

FUZZING ENGAGMENT PROCESS

- Reconnaissance
- Initial target selection
- Understanding the codebase
- Identifying critical code paths
- Writing and analyzing fuzzers
- Write and run the fuzzers
- Fix and improve fuzzers
- Triage running statistics and crashes
- Reporting and fixing vulnerabilities

FUZZFOREST RECONNAISSANCE

Usually hackers would be limited by their capacity to understand the codebase and identify critical code paths. Since GPT is excellent at understanding code and is not time constrained we simply extract all functions in the code.

```
repo_path = 'saved_repos'
sqlitedb = 'fuzzforest-gpt4.db'

libraries = {
    'tornado': 'https://github.com/tornadoweb/tornado',
    'django': 'https://github.com/django/django'}

LangFuzzRecon(sqlitedb, repo_path)
```

CREATING FUZZ TESTS

FuzzForest pulls the previously identified function information, combines it with our prompt and saves the created results in the DB.

```
langfuzz = LangFuzz(sqlitedb)
langfuzz.generate_fuzz_tests("django")

function_list.extend(langfuzz.get_functions_that_contain_string("django", 'parse'))
function_list.extend(langfuzz.get_functions_that_contain_string("django", 'load'))
function_list.extend(langfuzz.get_functions_by_filename("django", 'parse_xml'))

langfuzz.generate_fuzz_tests(library_name, function_list)
```

PROMPT

ATHERIS EXAMPLE CODE

+

ATHERIS API INFORMATION

+

COMPLEX ATHERIS EXAMPLE

+

IMPORTANT NOTES

+

WRITE A FUZZER FOR THE <FUNCTION_NAME> IN <LIBRARY_NAME>: <FUNCTION_CODE>

PROMPT GITHUB LINK: 100 LINES, 1300 TOKENS

PROMPT

```
# Python fuzz tests with atheris library https://github.com/google/atheris
# Generic atheris fuzz test example:
import atheris
import sys
import example_library
def TestOneInput(input_bytes):
  fdp = atheris.FuzzedDataProvider(input_bytes)
  data = fdp.ConsumeInt(sys.maxsize)
  example_library.CodeBeingFuzzed(data)
def main():
  atheris.Setup(sys.argv, TestOneInput)
  atheris.Fuzz()
if __name__ == "__main__":
  atheris.instrument_all() # This is needed for coverage to work.
  main()

When fuzzing Python, Atheris will report a failure if the Python code under test throws an uncaught
exception.
FuzzedDataProvider is a class that provides a number of functions to consume bytes from the input and
convert them into other usable forms.
Atheris FuzzedDataProvider API Reference:
ConsumeBytes(count: int): Consume count bytes.
ConsumeUnicode(count: int): Consume unicode characters. Might contain surrogate pair characters.
ConsumeUnicodeNoSurrogates(count: int): Consume unicode characters, but never generate surrogate pair
```

PROMPT

```
def main():
  if len(sys.argv) > 1 and sys.argv[1] == '--no_mutator':
    atheris.Setup(sys.argv, TestOneInput)
  else:
    atheris.Setup(sys.argv, TestOneInput, custom_mutator=CustomMutator)
  atheris.Fuzz()
if __name__ == "__main__":
    atheris.instrument_all()
    main()

IMPORTANT: ALWAYS use atheris.instrument_all() before calling main() to ensure that coverage is collected.
This is required for coverage to work. Also make sure all imported modules are using the exisiting and
correct attributes.
 def main():
    atheris.Setup(sys.argv, TestOneInput)
    atheris.Fuzz()

if __name__ == "__main__":
    atheris.instrument_all()
    main()
Important:
Before using any attributes, methods, or functions from libraries, make sure to check if they exist and
are accesible. Try to avoid triggering an AttributeError.


Write a fuzzer for
```

TWISTED.HTTP.PARSE_QS()

```python
import sys
import atheris
from twisted.web import http

def TestOneInput(data):
    fdp = atheris.FuzzedDataProvider(data)
    original = fdp.ConsumeUnicode(sys.maxsize)

    try:
        response = http.parse_qs(original)
        response.keys()
        response.values()
    except ValueError:
        None
    return

def main():
    atheris.Setup(sys.argv, TestOneInput)
    atheris.Fuzz()

if __name__ == "__main__":
    atheris.instrument_all()
    main()
```

BABEL.PARSE_DATE('4/1/04', LOCALE='EN_US')

```
import sys
import atheris
from babel.dates import parse_date
from babel.core import UnknownLocaleError

def TestOneInput(data):
    fdp = atheris.FuzzedDataProvider(data)
    try:
        string = fdp.ConsumeUnicode(sys.maxsize)
        locale = fdp.PickValueInList(['en_US', 'de_DE'])
        format = fdp.PickValueInList(['short', 'medium', 'long', 'full'])
        parse_date(string, locale=locale, format=format)
    except UnknownLocaleError:
        pass
    except ValueError:
        pass

def main():
    atheris.instrument_all()
    atheris.Setup(sys.argv, TestOneInput)
    atheris.Fuzz()
```

BOTO3.CLIENT('S3').DOWNLOAD_FILE(ARG1, ARG2, ARG3)

```
# Changes made:
# Added import re to validate Bucket name
# Added regex check to validate Bucket name before downloading the file"
def TestOneInput(input_bytes):
    s3 = boto3.client('s3')
    fdp = atheris.FuzzedDataProvider(input_bytes)
    Bucket = fdp.ConsumeUnicode(sys.maxsize)
    Key = fdp.ConsumeUnicode(sys.maxsize)
    Filename = fdp.ConsumeUnicode(sys.maxsize)
    if re.match("^[a-zA-Z0-9._]{1,255}$", Bucket):
        try:
            s3.download_file(Bucket, Key, Filename)
        except (ValueError, botocore.exceptions.ClientError):
            pass

def main():
    atheris.Setup(sys.argv, TestOneInput)
    atheris.Fuzz()

if __name__ == ""__main__"":
    atheris.instrument_imports()
    main()
```

CRYPTOGRAPHY DSA

```python
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import dsa, utils

def TestInput(input_bytes):
    fdp = atheris.FuzzedDataProvider(input_bytes)

    # Use different key sizes for broader coverage
    for key_size in [1024, 2048, 3072]:
        private_key = dsa.generate_private_key(key_size=key_size)
        public_key = private_key.public_key()

        data = fdp.ConsumeBytes(20)
        more_data = fdp.ConsumeBytes(20)

        # Try different hashing algorithms
        for hash_algorithm in [hashes.SHA1(), hashes.SHA256(), hashes.SHA512()]:
            hasher = hashes.Hash(hash_algorithm)
            hasher.update(data)
            hasher.update(more_data)
            digest = hasher.finalize()

            sig1 = private_key.sign(data, hash_algorithm)
            sig2 = private_key.sign(digest, utils.Prehashed(hash_algorithm))

            public_key.verify(sig1, data, hash_algorithm)

            # According to RFC 6979 (https://tools.ietf.org/html/rfc6979), the DSA signature is a pair of
            two integers usually referred to as (r, s).
            # This can be verified by decoding the signature, which is in DER format.
            r, s = utils.decode_dss_signature(sig2)
            assert r >= 0 and s >= 0
            public_key.verify(sig2, digest, utils.Prehashed(hash_algorithm))
```



SHOW US THE VULNERABILITIES!!

Library Name	Function Name	Bug Class
aiohttp	parse_http_date	OverflowError: Python int too large to convert to C long
ansible	parse_kv	UnicodeDecodeError: 'unicodeescape' codec can't decode bytes in position 0-9: illegal Unicode character
botocore	raw_config_parse	ValueError: embedded null byte
botocore	percent_encode_sequence	IndexError: string index out of range
botocore	json_decode_template_body	TypeError: argument of type 'int' is not iterable
django	normalize	IndexError: pop from empty list
django	parse_header_parameters	LookupError: unknown encoding:
flask	routes_command	SystemExit: 0
homeassistant	parse_time_expression	ZeroDivisionError: integer division or modulo by zero
numpy	parse_values	MemoryError: <EMPTY MESSAGE>
numpy	parse_values	OverflowError: cannot fit 'int' into an index-sized integer
numpy	norm	RuntimeError: Could not allocate list object!
PIL	decode_dxt1	DecompressionBombError: Image size exceeds limit of 178956970 pixels
pytorch	parse_yaml	UnicodeDecodeError: 'utf-8' codec can't decode byte 0xab in position 0: invalid start byte
rq	decode_redis_hash	InternalError: Unknown OpenSSL error.
scipy	parse_routine	OSError: could not read bytes
scipy	parse_ipp_file	UnboundLocalError: local variable 'fs' referenced before assignment
tornado	parse_config_file	ValueError: embedded null byte
twisted	loadAliasFile	AddressError: Parse error at b';' of (b';:', [b';', b':'])
yaml	parse_block_scalar_empty_line	UnicodeDecodeError: 'utf-8' codec can't decode byte 0x82 in position 1: invalid start byte

COMPLETE VULNERABILITY LIST: 72 FINDINGS TOTAL

SHOW US THE VULNERABILITIES!!

Library Name	Function Name	Bug Class
aiohttp	**parse_http_date**	**OverflowError: Python int too large to convert to C long**
ansible	**parse_kv**	**UnicodeDecodeError: 'unicodeescape' codec can't decode bytes in position 0-9: illegal Unicode character**
botocore	raw_config_parse	ValueError: embedded null byte
botocore	percent_encode_sequence	IndexError: string index out of range
botocore	json_decode_template_body	TypeError: argument of type 'int' is not iterable
django	normalize	IndexError: pop from empty list
django	parse_header_parameters	LookupError: unknown encoding:
flask	routes_command	SystemExit: 0
homeassistant	parse_time_expression	ZeroDivisionError: integer division or modulo by zero
numpy	parse_values	MemoryError: <EMPTY MESSAGE>
numpy	parse_values	OverflowError: cannot fit 'int' into an index-sized integer
numpy	norm	RuntimeError: Could not allocate list object!
PIL	decode_dxt1	DecompressionBombError: Image size exceeds limit of 178956970 pixels
pytorch	parse_yaml	UnicodeDecodeError: 'utf-8' codec can't decode byte 0xab in position 0: invalid start byte
rq	decode_redis_hash	InternalError: Unknown OpenSSL error.
scipy	parse_routine	OSError: could not read bytes
scipy	parse_ipp_file	UnboundLocalError: local variable 'fs' referenced before assignment
tornado	parse_config_file	ValueError: embedded null byte
twisted	loadAliasFile	AddressError: Parse error at b';' of (b';:', [b';', b':'])
yaml	parse_block_scalar_empty_line	UnicodeDecodeError: 'utf-8' codec can't decode byte 0x82 in position 1: invalid start byte

SHOW US THE VULNERABILITIES!!

Library Name	Function Name	Bug Class
aiohttp	parse_http_date	OverflowError: Python int too large to convert to C long
ansible	parse_kv	UnicodeDecodeError: 'unicodeescape' codec can't decode bytes in position 0-9: illegal Unicode character
botocore	raw_config_parse	ValueError: embedded null byte
botocore	percent_encode_sequence	IndexError: string index out of range
botocore	json_decode_template_body	TypeError: argument of type 'int' is not iterable
django	normalize	IndexError: pop from empty list
django	parse_header_parameters	LookupError: unknown encoding:
flask	**routes_command**	**SystemExit: 0**
homeassistant	parse_time_expression	ZeroDivisionError: integer division or modulo by zero
numpy	**parse_values**	**MemoryError: <EMPTY MESSAGE>**
numpy	parse_values	OverflowError: cannot fit 'int' into an index-sized integer
numpy	norm	RuntimeError: Could not allocate list object!
PIL	decode_dxt1	DecompressionBombError: Image size exceeds limit of 178956970 pixels
pytorch	parse_yaml	UnicodeDecodeError: 'utf-8' codec can't decode byte 0xab in position 0: invalid start byte
rq	decode_redis_hash	InternalError: Unknown OpenSSL error.
scipy	parse_routine	OSError: could not read bytes
scipy	parse_ipp_file	UnboundLocalError: local variable 'fs' referenced before assignment
tornado	parse_config_file	ValueError: embedded null byte
twisted	loadAliasFile	AddressError: Parse error at b';' of (b';:', [b';', b':'])
yaml	parse_block_scalar_empty_line	UnicodeDecodeError: 'utf-8' codec can't decode byte 0x82 in position 1: invalid start byte

SHOW US THE VULNERABILITIES!!

Library Name	Function Name	Bug Class
aiohttp	parse_http_date	OverflowError: Python int too large to convert to C long
ansible	parse_kv	UnicodeDecodeError: 'unicodeescape' codec can't decode bytes in position 0-9: illegal Unicode character
botocore	raw_config_parse	ValueError: embedded null byte
botocore	percent_encode_sequence	IndexError: string index out of range
botocore	json_decode_template_body	TypeError: argument of type 'int' is not iterable
django	normalize	IndexError: pop from empty list
django	parse_header_parameters	LookupError: unknown encoding:
flask	routes_command	SystemExit: 0
homeassistant	parse_time_expression	ZeroDivisionError: integer division or modulo by zero
numpy	parse_values	MemoryError: <EMPTY MESSAGE>
numpy	parse_values	OverflowError: cannot fit 'int' into an index-sized integer
numpy	norm	RuntimeError: Could not allocate list object!
PIL	**decode_dxt1**	**DecompressionBombError: Image size exceeds limit of 178956970 pixels**
pytorch	parse_yaml	UnicodeDecodeError: 'utf-8' codec can't decode byte 0xab in position 0: invalid start byte
rq	decode_redis_hash	InternalError: Unknown OpenSSL error.
scipy	parse_routine	OSError: could not read bytes
scipy	**parse_ipp_file**	**UnboundLocalError: local variable 'fs' referenced before assignment**
tornado	parse_config_file	ValueError: embedded null byte
twisted	loadAliasFile	AddressError: Parse error at b';' of (b';;', [b';', b':'])
yaml	parse_block_scalar_empty_line	UnicodeDecodeError: 'utf-8' codec can't decode byte 0x82 in position 1: invalid start byte

DATA ANALYSIS AND COST

GPT-3.5

20 LIBRARIES
365 CREATED FUZZERS
172 RUN=TRUE
47.12% RUN
$0.29 TO CREATE TESTS

FIX FUZZ TESTS
89/193 FIXED
71.51% WORK AFTER 5 TRY FIX 261/365
$0.42 COST TO FIX FUZZERS

$0.71 TOTAL COST
46 CRASHES



GPT-4

20 LIBRARIES
405 CREATED FUZZERS
121 RUN=TRUE
29.88% RUN
$18.74 TO CREATE TESTS

FIX FUZZ TESTS
228/284 FIXED
86.17% WORK AFTER 5 TRY FIX 349/405
$30.40 COST TO FIX FUZZERS

$49.14 TOTAL COST
65 CRASHES

OTHER MODELS

- Salesforce's CodeGen Mono 16B for Python
- Replit's 3B parameter model trained on 20 programming languages.
- BigCode's StarCoder on Hugging Face is a newer 16B parameter model trained on one trillion tokens sourced from 80+ programming languages, GitHub issues, Git commits, and Jupyter notebooks.

MODEL COMPARISON

SALESFORCE CODEGEN

```
@fuzzer()
def test_parse_values():
    input_strings = [
        '',
        '(', ')','', ',',
        'a*', 'a,', 'a, b', 'a,b', 'a, b c', 'a, b, c'
        ,
        '(a)', '(a,)', '(a,b)', '(a,b,)']
    output_strings = ['','','','a,a,a,a,a,a',
                  'a', 'a,b', 'a,b,c', 'a,b,c', 'a,b,c',
                  'a', 'a,b', 'a,b,c', 'a,b,c', 'a,b,c']
    for inp, out in zip(input_strings,output_strings):
        assert parse_values(inp) == out



def test_one_input(input_bytes):
  with open('/dev/urandom', 'rb') as urandom:
      seed = urandom.read(8)

  fuzzer = atheris.Atheris(seed)
  fuzzer.add_argument(input_bytes)
  fuzzer.fuzz()
```

BIGCODE STARCODER

```
def test_one_input(input_bytes):
  with open('/dev/urandom', 'rb') as urandom:
      seed = urandom.read(8)

  fuzzer = atheris.Atheris(seed)
  fuzzer.add_argument(input_bytes)
  fuzzer.fuzz()
```

NEXT STEPS IN DEVELOPMENT

- Add more languages, JavaScript is next.
- Research possibility of improving fuzzers, for example can we feed it existing chrome or linux kernel fuzzers and improve coverage?
- Integrate out of the box usage of different models.
- Improve coverage and analytics statistics.
- Unit Tests - write a unit test for the function which makes sure to catch this crash.
    - While performing this research Github announced Copilot X (no relation) which is supposed to integrate itself into a codebase and lists automatic unit test creation as a major feature.

IMPLICATION OF AI FOR SECURITY PEOPLE

- We keep hearing that ChatGPT will eliminate software devs, will it end security professionals too?
- I don’t believe so. There is increasing software creation and AI tools actually allow more people to write code than ever.
- More opportunities for bugs to be introduced and us to find them, though as has been the trend, bugs will become more subtle and harder to find.
- The Art, Science, and Engineering of Fuzzing
- We must live by the DEF CON theme and adapt to new tech and ride the wave into the future.

THANK YOU

In-depth Blog posts:

- https://infiniteforest.org/LLM+Fuzz/LLM+Fuzz+Part+1
- https://infiniteforest.org/LLM+Fuzz/LLM+Fuzz+Part+2

Atheris Fuzzing Engine:

- https://github.com/google/atheris

Good Prompt Course:

- https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/

Related Blogs:

- https://comby.dev/blog/2022/04/11/comby-decomposer-compiler-fuzzing
- https://martinfowler.com/articles/2023-chatgpt-xu-hao.html