CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Anota: Identifying Business Logic Vulnerabilities via Annotation-Based Sanitization

Meng Wang[1], Philipp Görz[1], Joschua Schilling[1], Keno Hassler[1], Liwei Guo[2], Thorsten Holz[3], Ali Abbasi[1]

[1]*CISPA Helmholtz Center for Information Security*
[2]*University of Electronic Science and Technology*
[3]*Max Planck Institute for Security and Privacy*

EMSEC

CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

UESTC 1956

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

Motivation Example

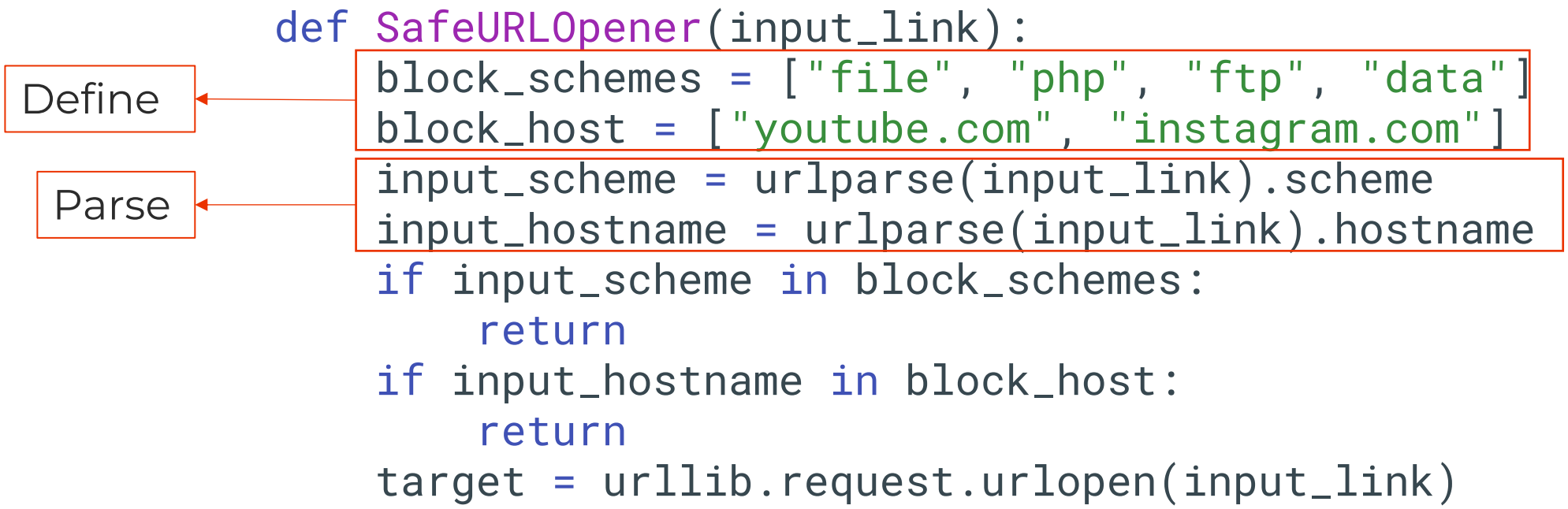
```
def SafeURLOpener(input_link):
    block_schemes = ["file", "php", "ftp", "data"]
    block_host = ["youtube.com", "instagram.com"]
    input_scheme = urlparse(input_link).scheme
    input_hostname = urlparse(input_link).hostname
    if input_scheme in block_schemes:
        return
    if input_hostname in block_host:
        return
    target = urllib.request.urlopen(input_link)
```

Motivation Example

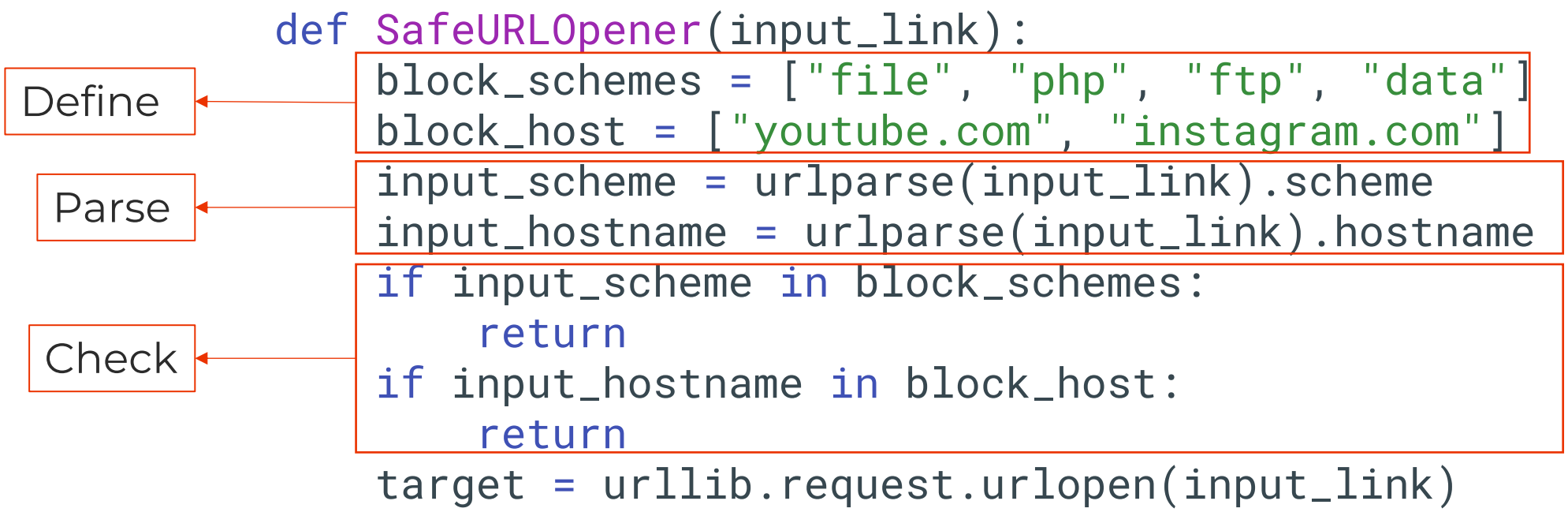
Define

```
def SafeURLOpener(input_link):
    block_schemes = ["file", "php", "ftp", "data"]
    block_host = ["youtube.com", "instagram.com"]
    input_scheme = urlparse(input_link).scheme
    input_hostname = urlparse(input_link).hostname
    if input_scheme in block_schemes:
        return
    if input_hostname in block_host:
        return
    target = urllib.request.urlopen(input_link)
```

Motivation Example



Motivation Example



Motivation Example

Define

Parse

Check

```
def SafeURLOpener(input_link):
    block_schemes = ["file", "php", "ftp", "data"]
    block_host = ["youtube.com", "instagram.com"]
    input_scheme = urlparse(input_link).scheme
    input_hostname = urlparse(input_link).hostname
    if input_scheme in block_schemes:
        return
    if input_hostname in block_host:
        return
    target = urllib.request.urlopen(input_link)
```

Intended behavior

Motivation Example

```
def SafeURLOpener(input_link):
    block_schemes = ["file", "php", "ftp", "data"]
    block_host = ["youtube.com", "instagram.com"]
    input_scheme = urlparse(input_link).scheme
    input_hostname = urlparse(input_link).hostname
    if input_scheme in block_schemes:
        return
    if input_hostname in block_host:
        return
    target = urllib.request.urlopen(input_link)
```

Define

Parse

Check

Intended behavior

CVE-2023-24329

Motivation Example

Define

Parse

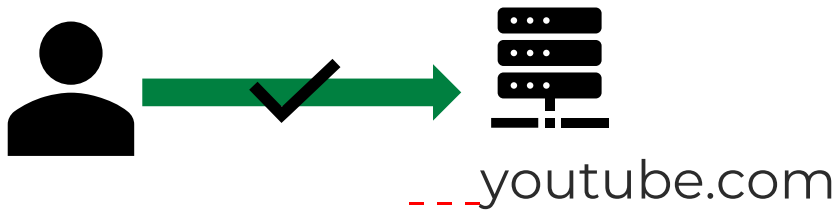
Check

```
def SafeURLOpener(input_link):
    block_schemes = ["file", "php", "ftp", "data"]
    block_host = ["youtube.com", "instagram.com"]
    input_scheme = urlparse(input_link).scheme
    input_hostname = urlparse(input_link).hostname
    if input_scheme in block_schemes:
        return
    if input_hostname in block_host:
        return
    target = urllib.request.urlopen(input_link)
```

Intended behavior

CVE-2023-24329

Unintended behavior



The Blindspot: Business Logic Vulnerabilities

- Business Logic Vulnerabilities
  - Caused by flaws in the design and implementation
  - Abuse legitimate functionality
  - Application specific and context dependent

Business Logic Vulnerabilities in CWE Top 40

Rank	CWE ID	Name
5	22	Path Traversal
9	962	Missing Authorization
10	434	Unrestricted Upload of File
11	94	Code Injection
14	287	Improper Authentication
15	269	Improper Privilege Management
16	502	Deserialization of Untrusted Data
17	200	Exposure of Sensitive Information
18	863	Incorrect Authorization
25	306	Missing Authentication for Critical Functions
27	668	Exposure of Resource to Wrong Sphere
29	427	Uncontrolled Search Path Element
30	639	Authorization Bypass
31	532	Insertion of Sensitive Information into Log File
32	732	Incorrect Permission Assignment
35	522	Insufficiently Protected Credentials
37	203	Observable Discrepancy
...	...	...

- 27 of CWE Top 40
- Heuristic-based sanitizers
  - Atropos[1]
  - ...

1 (SEC '24) Atropos: effective fuzzing of web applications for server-side vulnerabilities.

Business Logic Vulnerabilities in CWE Top 40

Rank	CWE ID	Name
5	22	Path Traversal
9	962	Missing Authorization
10	434	Unrestricted Upload of File
11	94	Code Injection
14	287	Improper Authentication
15	269	Improper Privilege Management
16	502	Deserialization of Untrusted Data
17	200	Exposure of Sensitive Information
18	863	Incorrect Authorization
25	306	Missing Authentication for Critical Functions
27	668	Exposure of Resource to Wrong Sphere
29	427	Uncontrolled Search Path Element
30	639	Authorization Bypass
31	532	Insertion of Sensitive Information into Log File
32	732	Incorrect Permission Assignment
35	522	Insufficiently Protected Credentials
37	203	Observable Discrepancy
...	...	...

- 27 of CWE Top 40
- Heuristic-based sanitizers
  - Atropos[1]
  - ...

Automated tools lack **Semantic Context**.

1 (SEC '24) Atropos: effective fuzzing of web applications for server-side vulnerabilities.

Business Logic Vulnerabilities in CWE Top 40

Rank	CWE ID	Name
5	22	Path Traversal
9	962	Missing Authorization
10	434	Unrestricted Upload of File
11	94	Code Injection
14	287	Improper Authentication
15	269	Improper Privilege Management
16	502	Deserialization of Untrusted Data
17	200	Exposure of Sensitive Information
18	863	Incorrect Authorization
25	306	Missing Authentication for Critical Functions
27	668	Exposure of Resource to Wrong Sphere
29	427	Uncontrolled Search Path Element
30	639	Authorization Bypass
31	532	Insertion of Sensitive Information into Log File
32	732	Incorrect Permission Assignment
35	522	Insufficiently Protected Credentials
37	203	Observable Discrepancy
…	…	…

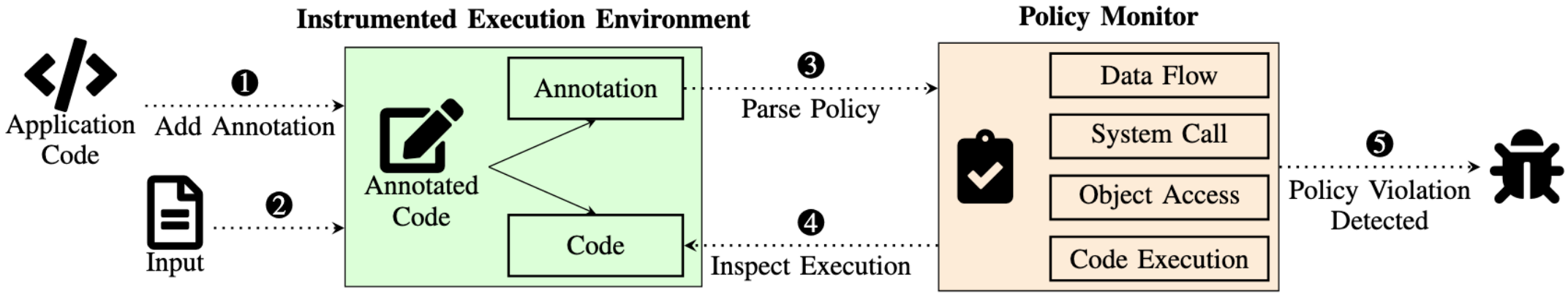
- 27 of CWE Top 40
- Heuristic-based sanitizers
  - Atropos[1]
  - …

Automated tools lack **Semantic Context**.

Trade generability for automation

1 (SEC '24) Atropos: effective fuzzing of web applications for server-side vulnerabilities.

Our Solution: ANOTA



- Power of human + machine
- Framework
  - Annotations (Human): implicit human insights -> explicit machine verifiable policies
  - Policy Monitor (Machine): Monitors execution

Annotation Design

Annotation Design

CWE ID	Name
22	Path Traversal
434	Unrestricted Upload of File
918	Server-Side Request Forgery
...	...

Unintended System Call Usage

Annotation Design

CWE ID	Name
22	Path Traversal
434	Unrestricted Upload of File
918	Server-Side Request Forgery
...	...

Unintended System Call Usage

CWE ID	Name
200	Exposure of Sensitive Information
532	Insertion of Sensitive Information into Log File
522	Insufficiently Protected Credentials
...	...

Unintended Data Flow

Annotation Design

CWE ID	Name
22	Path Traversal
434	Unrestricted Upload of File
918	Server-Side Request Forgery
...	...

Unintended System Call Usage

CWE ID	Name
862	Missing Authorization
287	Improper Authentication
269	Improper Privilege Management
...	...

Unintended Code Execution

CWE ID	Name
200	Exposure of Sensitive Information
532	Insertion of Sensitive Information into Log File
522	Insufficiently Protected Credentials
...	...

Unintended Data Flow

Annotation Design

CWE ID	Name
22	Path Traversal
434	Unrestricted Upload of File
918	Server-Side Request Forgery
...	...

Unintended System Call Usage

CWE ID	Name
862	Missing Authorization
287	Improper Authentication
269	Improper Privilege Management
...	...

Unintended Code Execution

CWE ID	Name
200	Exposure of Sensitive Information
532	Insertion of Sensitive Information into Log File
522	Insufficiently Protected Credentials
...	...

Unintended Data Flow

CWE ID	Name
862	Missing Authorization
522	Insufficiently Protected Credentials
668	Exposure of Resource to Wrong Sphere
...	...

Unintended Object Access

Annotation Examples

Annotation Examples

```
1  def SafeURLOpener(inputLink):
2  ...
3      SYSCALL.NETWORK.BLOCK.SCHEME("file","php"...)
4      SYSCALL.NETWORK.BLOCK.HOST("youtube.com"...)
5      target = urllib.request.urlopen(inputLink)
6      print(target.read())
7      SYSCALL.NETWORK.CLEAR()
```

Annotated Motivation Example

Annotation Examples

```
1  def SafeURLOpener(inputLink):
2  ...
3      SYSCALL.NETWORK.BLOCK.SCHEME("file","php"...)
4      SYSCALL.NETWORK.BLOCK.HOST("youtube.com"...)
5      target = urllib.request.urlopen(inputLink)
6      print(target.read())
7      SYSCALL.NETWORK.CLEAR()
```

Annotated Motivation Example

```
1  ...
2  @route("/static/<filename>")
3  def static(self, name, filetype, filename):
4      SYSCALL.FILE.ALLOW(os.path.join(cwd, "static"))
5      return send_from_directory(f"{cwd}/static",
           filename)
6  ...
```

System Call Annotation

Annotation Examples

```
def SafeURLOpener(inputLink):
...
    SYSCALL.NETWORK.BLOCK.SCHEME("file","php"...)
    SYSCALL.NETWORK.BLOCK.HOST("youtube.com"...)
    target = urllib.request.urlopen(inputLink)
    print(target.read())
    SYSCALL.NETWORK.CLEAR()
```

Annotated Motivation Example

```
...
@route("/static/<filename>")
def static(self, name, filetype, filename):
    SYSCALL.FILE.ALLOW(os.path.join(cwd, "static"))
    return send_from_directory(f"{cwd}/static",
        filename)
...
```

System Call Annotation

```
...
user_credential = get_credential(user_id)
TAINT(user_credential, sanitization=[hash])
hashed_credential = hash(user_credential)
logfile.write("credential %s collected with hash %s\n"
    , user_credential, hashed_credential)
...
```

Data Flow Annotation

Annotation Examples

```
def SafeURLOpener(inputLink):
...
    SYSCALL.NETWORK.BLOCK.SCHEME("file","php"...)
    SYSCALL.NETWORK.BLOCK.HOST("youtube.com"...)
    target = urllib.request.urlopen(inputLink)
    print(target.read())
    SYSCALL.NETWORK.CLEAR()
```

Annotated Motivation Example

```
...
@route("/static/<filename>")
def static(self, name, filetype, filename):
    SYSCALL.FILE.ALLOW(os.path.join(cwd, "static"))
    return send_from_directory(f"{cwd}/static",
        filename)
...
```

System Call Annotation

```
...
user_credential = get_credential(user_id)
TAINT(user_credential, sanitization=[hash])
hashed_credential = hash(user_credential)
logfile.write("credential %s collected with hash %s\n"
    , user_credential, hashed_credential)
...
```

Data Flow Annotation

```
...
if is_auth:
    Execution.BLOCK()
    user_list.append(new_user)

WATCH.Allow(user_list, "r")
user_list.remove(existing_user) if is_auth else print(
    "not authenticated")
...
```

Code Execution/Object Access Annotation

Evaluation I: Rediscovering Known Bugs with Fuzzing

Evaluation I: Rediscovering Known Bugs with Fuzzing

Is the annotation system expressive enough?

Evaluation I: Rediscovering Known Bugs with Fuzzing

Is the annotation system expressive enough?

ID	Name	CWE	Stars /10^3	Vuln. Identifier
1	internetarchive	362	1.5	SNYK-6141253
2	b2-sdk-python	362	0.2	CVE-2022-23651
3	B2_Command_Line_Tool	362	0.5	CVE-2022-23653
4	langchain	918	80.1	CVE-2024-2057
5	langchain	918	80.1	CVE-2024-0243
6	label-studio	918	16.1	CVE-2023-47116
7	whoogle-search	918	8.7	CVE-2024-22205
8	gradio	918	27.8	SNYK-6141123
9	Paddle	22	21.5	CVE-2024-0818
10	xtts-api-server	22	0.2	SNYK-6398416
11	langchain	22	80.1	CVE-2024-28088
12	esphome	22	7.4	CVE-2024-27081
12	onnx	22	16.6	CVE-2024-27318
13	label-studio	434	16.1	SNYK-6347239
14	zenml	434	3.6	CVE-2024-28424
15	inventree	434	3.6	CVE-2022-2111
16	GibsonEnv	502	0.8	CVE-2024-0959
17	Transformers	502	123.0	CVE-2023-6730
18	synthcity	502	0.3	CVE-2024-0936
19	Apache-Airflow	862	34.0	CVE-2023-50944
20	changedetection.io	306	14.6	CVE-2024-23329
21	aries-cloudagent	287	0.4	CVE-2024-21669
22	MobSF	276	16.1	CVE-2023-42261
23	mlflow	287	17.1	CVE-2023-6014
24	calibre-web	863	11.3	CVE-2022-0405
25	wagtail	200	17.1	CVE-2023-45809
26	ansible-core	20	60.7	CVE-2024-0690
27	pyLoad	200	3.1	CVE-2024-21644
28	omise	200	0.0	SNYK-6138437
29	airflow-providers-celery	200	34.0	CVE-2023-46215
30	horizon	601	1.3	CVE-2020-29565
31	evennia	601	1.7	SNYK-6591326
32	pyLoad	601	3.1	CVE-2024-24808
33	llama_index	94	31.7	CVE-2024-3098
34	Aim	94	4.8	CVE-2024-2195
35	vantage6	94	0.0	CVE-2024-21649
36	DIRAC	668	0.1	CVE-2024-29905
37	fonttools	611	4.1	CVE-2023-45139
38	OWSLib	611	0.4	CVE-2023-27476
39	untangle	611	0.6	CVE-2022-31471
40	AccessControl	269	0.0	CVE-2024-51734
41	Jupyter_Core	427	0.2	CVE-2022-39286
42	clearml	522	0.0	CVE-2024-24595
43	indico	639	0.0	CVE-2024-50633
44	vantage6	287	0.0	CVE-2024-21653
45	Apache-Superset	287	57.8	CVE-2023-27526
46	zenml	287	3.6	CVE-2024-25723
47	nautobot-device-onboarding	200	0.0	CVE-2023-48700

Evaluation I: Rediscovering Known Bugs with Fuzzing

Is the annotation system expressive enough?

43/47 Identified

4 failures

ID	Name	CWE	Stars /10^3	Vuln. Identifier
1	internetarchive	362	1.5	SNYK-6141253
2	b2-sdk-python	362	0.2	CVE-2022-23651
3	B2_Command_Line_Tool	362	0.5	CVE-2022-23653
4	langchain	918	80.1	CVE-2024-2057
5	langchain	918	80.1	CVE-2024-0243
6	label-studio	918	16.1	CVE-2023-47116
7	whoogle-search	918	8.7	CVE-2024-22205
8	gradio	918	27.8	SNYK-6141123
9	Paddle	22	21.5	CVE-2024-0818
10	xtts-api-server	22	0.2	SNYK-6398416
11	langchain	22	80.1	CVE-2024-28088
12	esphome	22	7.4	CVE-2024-27081
12	onnx	22	16.6	CVE-2024-27318
13	label-studio	434	16.1	SNYK-6347239
14	zenml	434	3.6	CVE-2024-28424
15	inventree	434	3.6	CVE-2022-2111
16	GibsonEnv	502	0.8	CVE-2024-0959
17	Transformers	502	123.0	CVE-2023-6730
18	synthcity	502	0.3	CVE-2024-0936
19	Apache-Airflow	862	34.0	CVE-2023-50944
20	changedetection.io	306	14.6	CVE-2024-23329
21	aries-cloudagent	287	0.4	CVE-2024-21669
22	MobSF	276	16.1	CVE-2023-42261
23	mlflow	287	17.1	CVE-2023-6014
24	calibre-web	863	11.3	CVE-2022-0405
25	wagtail	200	17.1	CVE-2023-45809
26	ansible-core	20	60.7	CVE-2024-0690
27	pyLoad	200	3.1	CVE-2024-21644
28	omise	200	0.0	SNYK-6138437
29	airflow-providers-celery	200	34.0	CVE-2023-46215
30	horizon	601	1.3	CVE-2020-29565
31	evennia	601	1.7	SNYK-6591326
32	pyLoad	601	3.1	CVE-2024-24808
33	llama_index	94	31.7	CVE-2024-3098
34	Aim	94	4.8	CVE-2024-2195
35	vantage6	94	0.0	CVE-2024-21649
36	DIRAC	668	0.1	CVE-2024-29905
37	fonttools	611	4.1	CVE-2023-45139
38	OWSLib	611	0.4	CVE-2023-27476
39	untangle	611	0.6	CVE-2022-31471
40	AccessControl	269	0.0	CVE-2024-51734
41	Jupyter_Core	427	0.2	CVE-2022-39286
42	clearml	522	0.0	CVE-2024-24595
43	indico	639	0.0	CVE-2024-50633
44	vantage6	287	0.0	CVE-2024-21653
45	Apache-Superset	287	57.8	CVE-2023-27526
46	zenml	287	3.6	CVE-2024-25723
47	nautobot-device-onboarding	200	0.0	CVE-2023-48700

Evaluation I: Rediscovering Known Bugs with Fuzzing

Is the annotation system expressive enough?

43/47 Identified

4 failures

- Find an input bypass the patch

ID	Name	CWE	Stars /10^3	Vuln. Identifier
1	internetarchive	362	1.5	SNYK-6141253
2	b2-sdk-python	362	0.2	CVE-2022-23651
3	B2_Command_Line_Tool	362	0.5	CVE-2022-23653
4	langchain	918	80.1	CVE-2024-2057
5	langchain	918	80.1	CVE-2024-0243
6	label-studio	918	16.1	CVE-2023-47116
7	whoogle-search	918	8.7	CVE-2024-22205
8	gradio	918	27.8	SNYK-6141123
9	Paddle	22	21.5	CVE-2024-0818
10	xtts-api-server	22	0.2	SNYK-6398416
11	langchain	22	80.1	CVE-2024-28088
12	esphome	22	7.4	CVE-2024-27081
12	onnx	22	16.6	CVE-2024-27318
13	label-studio	434	16.1	SNYK-6347239
14	zenml	434	3.6	CVE-2024-28424
15	inventree	434	3.6	CVE-2022-2111
16	GibsonEnv	502	0.8	CVE-2024-0959
17	Transformers	502	123.0	CVE-2023-6730
18	synthcity	502	0.3	CVE-2024-0936
19	Apache-Airflow	862	34.0	CVE-2023-50944
20	changedetection.io	306	14.6	CVE-2024-23329
21	aries-cloudagent	287	0.4	CVE-2024-21669
22	MobSF	276	16.1	CVE-2023-42261
23	mlflow	287	17.1	CVE-2023-6014
24	calibre-web	863	11.3	CVE-2022-0405
25	wagtail	200	17.1	CVE-2023-45809
26	ansible-core	20	60.7	CVE-2024-0690
27	pyLoad	200	3.1	CVE-2024-21644
28	omise	200	0.0	SNYK-6138437
29	airflow-providers-celery	200	34.0	CVE-2023-46215
30	horizon	601	1.3	CVE-2020-29565
31	evennia	601	1.7	SNYK-6591326
32	pyLoad	601	3.1	CVE-2024-24808
33	llama_index	94	31.7	CVE-2024-3098
34	Aim	94	4.8	CVE-2024-2195
35	vantage6	94	0.0	CVE-2024-21649
36	DIRAC	668	0.1	CVE-2024-29905
37	fonttools	611	4.1	CVE-2023-45139
38	OWSLib	611	0.4	CVE-2023-27476
39	untangle	611	0.6	CVE-2022-31471
40	AccessControl	269	0.0	CVE-2024-51734
41	Jupyter_Core	427	0.2	CVE-2022-39286
42	clearml	522	0.0	CVE-2024-24595
43	indico	639	0.0	CVE-2024-50633
44	vantage6	287	0.0	CVE-2024-21653
45	Apache-Superset	287	57.8	CVE-2023-27526
46	zenml	287	3.6	CVE-2024-25723
47	nautobot-device-onboarding	200	0.0	CVE-2023-48700

Evaluation II: Finding Zero-Day Bugs with Fuzzing

Evaluation II: Finding Zero-Day Bugs with Fuzzing

- **Targets:** actively maintained projects

Evaluation II: Finding Zero-Day Bugs with Fuzzing

- **Targets:** actively maintained projects

ID	Name	CWE	Stars /10^3	LoC /10^3	Type	Time /min
1	OpenAgents	434	3.3	13	SC	60
2	ihatemoney	732	1.1	9	OA	30
3	Home Assistant core	532	68.0	1629	DF	70
4	Apache-Airflow	367	33.9	658	DF	90
5	FileCodeBox	532	3.0	5	DF	45
6	nebari	532	0.3	14	DF	50
7	SolidUI	532	0.5	5	DF	75
8	WordOps	532	1.2	14	DF	40
9	WordOps	367	1.2	14	DF	40
10	ArchiveBox	367	19.3	13	DF	20
11	Apache-Superset	434	57.5	175	SC	80
12	cmdb	434	1.2	20	SC	40
13	zenml	367	3.6	190	DF	65
14	MemGPT	208	8.6	25	DF	40
15	pyspider	208	16.3	15	DF	55
16	alexa_media_player	532	1.3	7	DF	30
17	comfyui_controlnet_aux	94	1.4	196	SC	70
18	lithops	94	0.3	29	SC	45
19	Linly-Talker	94	0.7	37	SC	50
20	cmssw	94	1.1	1597	SC	95
21	Microsoft RecAI	94	0.4	29	SC	45
22	calibre-web	434	11.3	30	SC	85

Evaluation II: Finding Zero-Day Bugs with Fuzzing

- **Targets:** actively maintained projects

- 4 with active bug bounties
- 1 with annual security audit
- **22** New zero-days
- **17** CVEs assigned

ID	Name	CWE	Stars /10^3	LoC /10^3	Type	Time /min
1	OpenAgents	434	3.3	13	SC	60
2	ihatemoney	732	1.1	9	OA	30
3	Home Assistant core	532	68.0	1629	DF	70
4	Apache-Airflow	367	33.9	658	DF	90
5	FileCodeBox	532	3.0	5	DF	45
6	nebari	532	0.3	14	DF	50
7	SolidUI	532	0.5	5	DF	75
8	WordOps	532	1.2	14	DF	40
9	WordOps	367	1.2	14	DF	40
10	ArchiveBox	367	19.3	13	DF	20
11	Apache-Superset	434	57.5	175	SC	80
12	cmdb	434	1.2	20	SC	40
13	zenml	367	3.6	190	DF	65
14	MemGPT	208	8.6	25	DF	40
15	pyspider	208	16.3	15	DF	55
16	alexa_media_player	532	1.3	7	DF	30
17	comfyui_controlnet_aux	94	1.4	196	SC	70
18	lithops	94	0.3	29	SC	45
19	Linly-Talker	94	0.7	37	SC	50
20	cmssw	94	1.1	1597	SC	95
21	Microsoft RecAI	94	0.4	29	SC	45
22	calibre-web	434	11.3	30	SC	85

Evaluation III: Annotation Study

ID	Gradio *(SSRF)*	xtts-api-server *(Path Traversal)*	cmdb *(Unrestricted File Upload)*	temporai *(Untrusted Deserialization)*	WordOps *(Information Leakage)*	changedetection.io *(Broken Access Control)*
P1	✓	✓	✓	✓	✓	✗
P2	✓	✓	✓	✓	✓	✗
P3	✓	✓	✓	✓	✓	✓
P4	✓	✓	✓	✓	✓	✓
P5	✓	✓	✓	✓	✓	✗
P6	✓	✓	✓	✓	✓	✓
P7	✓	✓	✓	✓	✓	✓
P8	✓	✓	✓	✓	✓	✗
P9	✓	✗	✓	✓	✓	✗
P10	✓	✓	✗	✓	✗	✗
P11	✗	✓	✓	✓	✓	✗

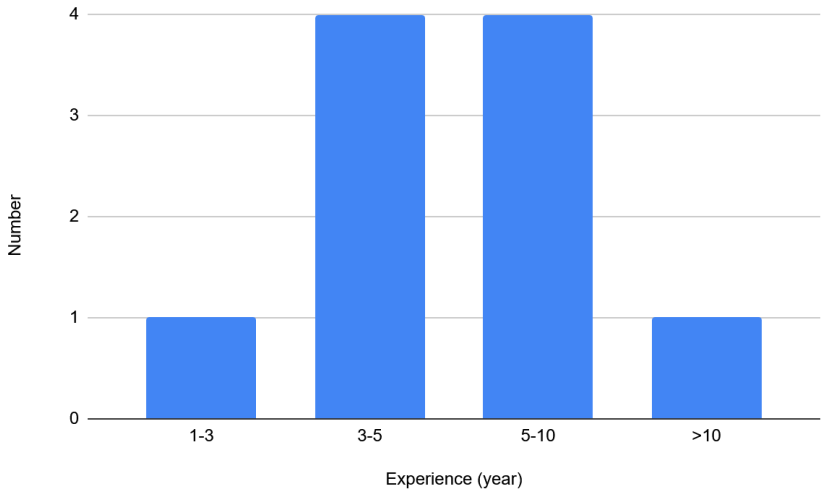
- 11 participants without prior knowledge
  - 83.3% success rate
  - About 60 minutes to annotate an application

Evaluation IV: Real-world Developer Study

1. How many years of software development experience do you have?
2. Have you dealt with business-logic vulnerabilities?
3. How do you currently discover or test for business-logic vulnerabilities?
4. What do you see as the biggest potential obstacles to adopting such a tool in your project?

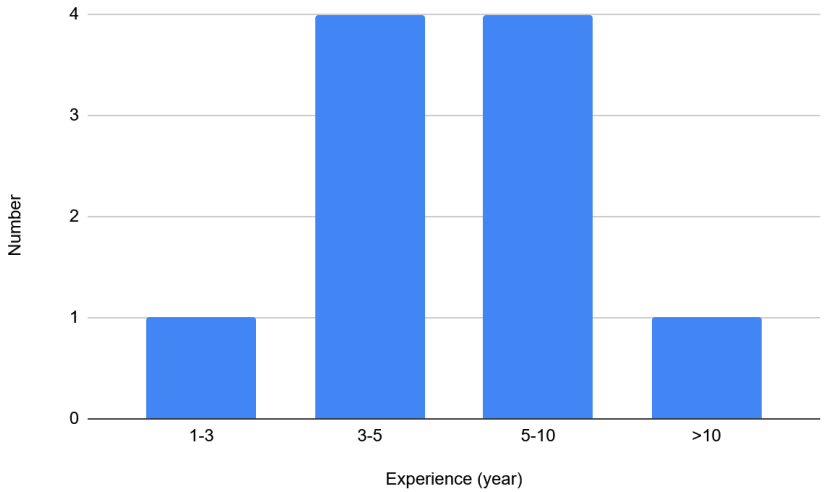
Evaluation IV: Real-world Developer Study

1. How many years of software development experience do you have?
2. Have you dealt with business-logic vulnerabilities?
3. How do you currently discover or test for business-logic vulnerabilities?
4. What do you see as the biggest potential obstacles to adopting such a tool in your project?



Evaluation IV: Real-world Developer Study

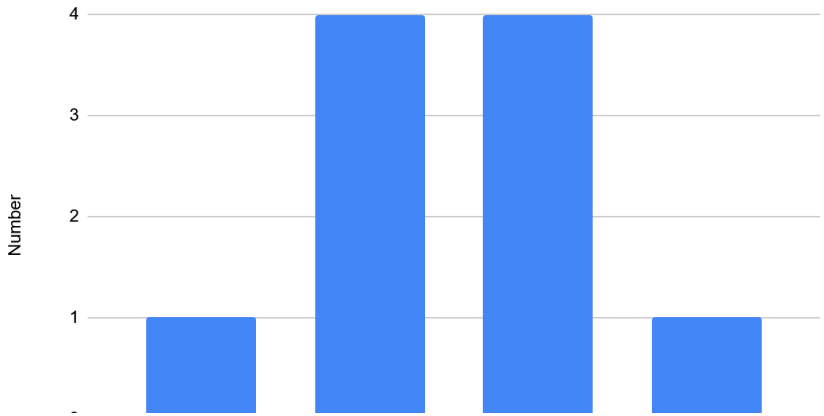
1. How many years of software development experience do you have?
2. Have you dealt with business-logic vulnerabilities?
3. How do you currently discover or test for business-logic vulnerabilities?
4. What do you see as the biggest potential obstacles to adopting such a tool in your project?

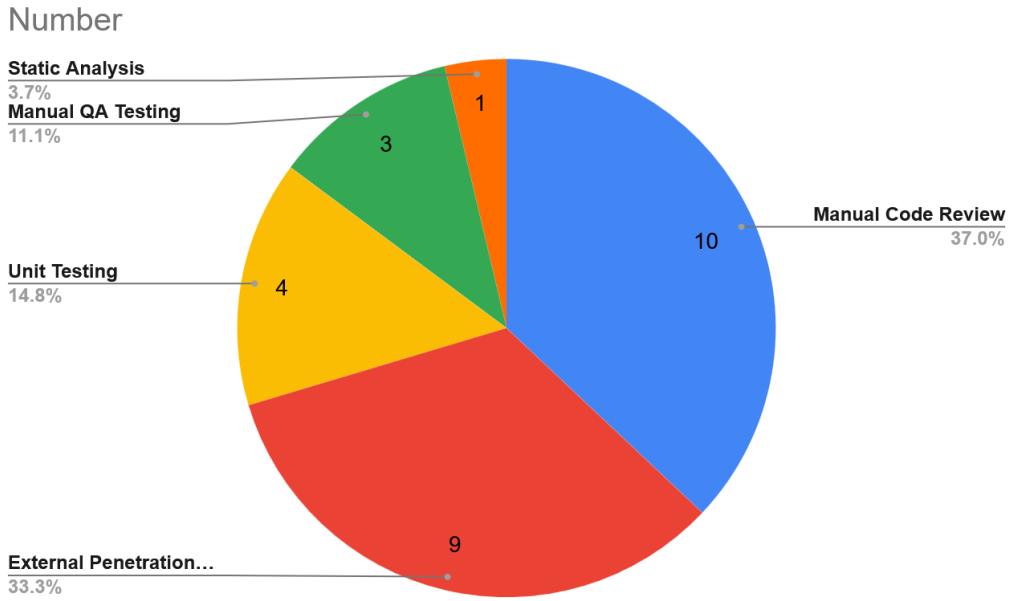


10/10 have encountered business logic vulnerabilities

Evaluation IV: Real-world Developer Study

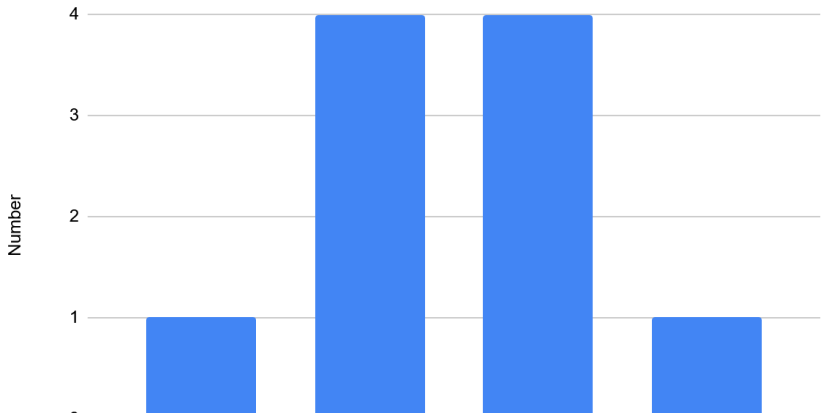
1. How many years of software development experience do you have?
2. Have you dealt with business-logic vulnerabilities?
3. How do you currently discover or test for business-logic vulnerabilities?
4. What do you see as the biggest potential obstacles to adopting such a tool in your project?



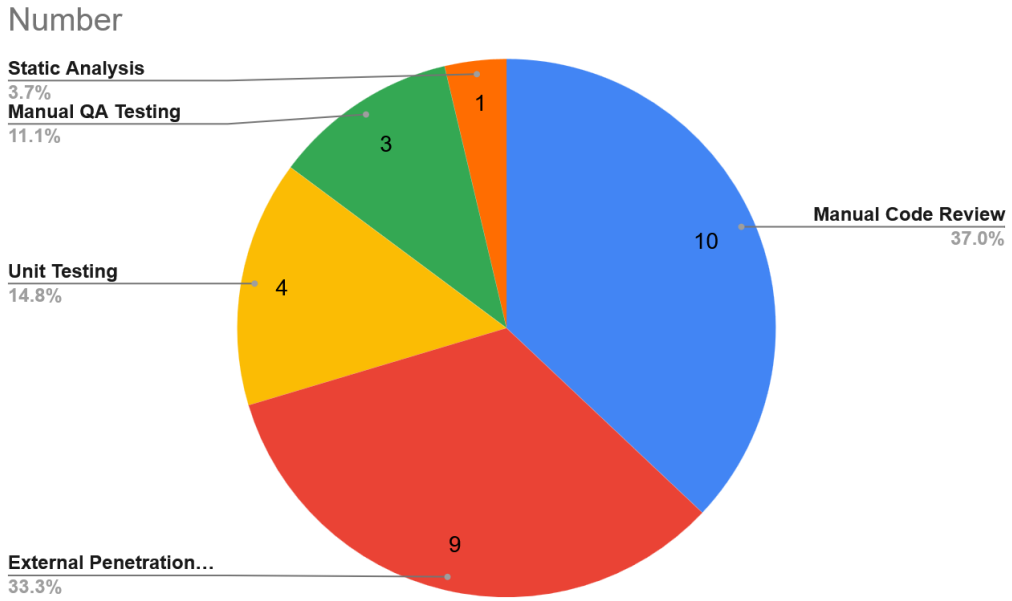


Evaluation IV: Real-world Developer Study

1. How many years of software development experience do you have?
2. Have you dealt with business-logic vulnerabilities?
3. How do you currently discover or test for business-logic vulnerabilities?
4. What do you see as the biggest potential obstacles to adopting such a tool in your project?



- Obstacles
  - Learning curve (10/10)
  - Tool integration (6/10)
  - Code readability (2/10)



Conclusion

- **Context is Needed:** Business logic vulnerabilities are invisible to automated tools without understanding intent.
- **The Paradigm Shift:** We move from *inferring* context with brittle heuristics to *enforcing* explicitly defined security policies.
- **Empowering Fuzzing:** ANOTA enables fuzzers to detect **43** known vulnerabilities and **22** zero-day vulnerabilities (17 CVEs assigned).

Questions & answers