

# Analisi del Tool FlashSyn e classificazione delle vulnerabilità trovate

---

Studente: Migliaccio Luca  
Matricola: M63001573

Professore: Natella Roberto

# Introduzione

- **Primo obiettivo:** riprodurre una o più righe della Tabella 3 presente nel paper «*FlashSyn: Flash Loan Attack via Counter Example Driven Approximation (ICSE 2024)*»

- **Caratteristiche dell'ambiente**

- **OS:** Windows 11 Home
- **Processore:** AMD Ryzen 7 5825U with Radeon Graphics, 2.00 GHz
- **RAM:** 16 GB
- **Architettura:** 64-bit OS, x64-based processor

Benchmark	FlashSyn-poly								FlashSyn-inter			Precise
	AC	AP	GL	GP	IDP	TDP	Profit	Time	TDP	Profit	Time	Profit
bZx1	3	3	2	1194	5192	5849	2392	422	6373	2302 <sup>†</sup>	441	cs
Harvest_USDT	4	4	4	338448	8000	9325	110139 <sup>†</sup>	670	10289	86798 <sup>†</sup>	7579	cs
Harvest_USDC	4	4	4	307416	8000	8912	59614 <sup>†</sup>	677	10914	110051 <sup>†</sup>	8349	cs
Eminence	4	4	5	1674278	8000	8780	1507174	1191	8104	/	/	1606965
ValueDeFi	6	6	6	8618002	12000	19975	8378194 <sup>†</sup>	4691	15758	6428341 <sup>†</sup>	11089	cx
CheeseBank	8	3	8	3270347	2679	2937	1946291 <sup>†</sup>	4391	2715	1101547 <sup>†</sup>	10942	2816762 <sup>†</sup>
Warp	6	3	6	1693523	6000	6000	2773345 <sup>†</sup>	1164	6000	/	/	2645640 <sup>†</sup>
bEarnFi	2	2	4	18077	4000	4854	13770	470	4652	12329	688	13832
AutoShark	8	3	8	1381	2753	2753	1372 <sup>†</sup>	5484	2753	/	/	cx
ElevenFi	5	2	5	129741	4000	4070	129658	409	4326	85811	898	cx
ApeRocket	7	3	6	1345	6000	6402	1333 <sup>†</sup>	733	6235	1037 <sup>†</sup>	3238	cs
Wdoge	5	1	5	78	2000	2001	75	272	2080	75	289	75
Novo	4	2	4	24857	4000	4164	20210	702	4031	23084	861	cx
OneRing	2	2	2	1534752	4000	4710	1814882	585	4218	1942188	367	cx
Puppet	3	3	2	89000	6000	6301	89000 <sup>†</sup>	1203	6452	87266 <sup>†</sup>	1238	89000 <sup>†</sup>
PuppetV2	4	3	3	953100	4491	4836	747799 <sup>†</sup>	2441	5061	362541 <sup>†</sup>	2835	647894 <sup>†</sup>
						Solved:16/18 Avg. Time: 1594			Solved:13/18 Avg. Time: 3754			

**N.B.** è stato utilizzato **WSL** (con Ubuntu 22.04).

# Concetti chiave (1/2)

- **Flash Loan:** intendiamo un «*prestito istantaneo*», ovvero si prende un'enorme quantità di fondi senza garanzie, ma si deve ripagare il tutto all'interno della medesima transazione. Questo tipo di prestito mostra le seguenti caratteristiche:
  - *nessuna garanzia richiesta*: non si deve fornire un **collaterale** per ottenere il prestito
  - *durata di una singola transazione*: il prestito viene **concesso, usato e rimborsato nella stessa transazione blockchain**

**N.B.** se entro la fine della transazione il prestito non viene ripagato, l'intera transazione fallisce (viene annullata). Questo avviene grazie alla proprietà fondamentale delle blockchain di **atomicità delle transazioni**.

- **DeFi:** intendiamo «*finanza decentralizzata*», ed è un insieme di servizi e strumenti finanziari costruiti su **blockchain pubbliche** (principalmente Ethereum) che permettono di svolgere attività tipiche del sistema finanziario tradizionale (come prestiti, scambi, investimenti, assicurazioni) senza intermediari centralizzati come banche o broker.

# Concetti chiave (2/2)

- **Flash Loan Attack:** intendiamo un tipo di *minaccia alla sicurezza* che sfrutta i difetti di progettazione dei contratti DeFi → consiste in una sequenza di azioni in cui l'aggressore prende in prestito una grande somma e la utilizza per sfruttare le vulnerabilità dello smart contract.
- **FlashSyn:** è un tool che simula in maniera automatica attacchi flash loan su protocolli DeFi, partendo solo dal codice del protocollo, senza la necessità di sapere cosa fa esattamente ogni funzione. In particolare, esplora automaticamente *tutte le possibili azioni* che un attaccante potrebbe fare.

Le *tecniche chiave* che utilizza sono due:

1. **Synthesis via Approximation** → invece di utilizzare il codice sorgente, raccoglie dati di input-output da varie chiamate di funzione. Il comportamento delle funzioni viene poi approssimato tramite **regressione polinomiale (FlashSyn-poly)** oppure **interpolazione basata sui nearest-neighbor (FlashSyn-inter)**.
2. **Counterexample-Driven Refinement** → se un attacco che è stato sintetizzato «non funziona bene», allora viene registrato come **controesempio**. Questo permette a FlashSyn di migliorare le approssimazioni.

**N.B.** questo tool cerca la *sequenza di azioni (attacco)* con cui si può ottenere il **massimo profitto**. Successivamente, verifica se quell'*attacco sintetizzato* funziona davvero sulla blockchain (simulata): se i risultati sono imprecisi, affina i modelli e riprova.

# Osservazioni

- FlashSyn utilizza un approccio di **analisi statica combinata con sintesi di sequenze di chiamate (symbolic execution)** per individuare vulnerabilità nei contratti DeFi.
  - Esamina il *codice sorgente* o *bytecode* senza eseguirlo.
  - Identifica percorsi critici e condizioni di errore (es. overflow, chiamate non autorizzate).
- Anche se non è un fuzzer classico, FlashSyn esplora lo spazio degli input tramite sintesi simbolica, generando migliaia di sequenze in grado di "*stressare*" il contratto.
- Il comportamento anomalo viene poi misurato in termini di **profitto ottenuto**, come metrica di rischio.

# Steps per eseguire FlashSyn in Docker (1/2)

1. Installare **Docker** (opzionale se già presente sul sistema)
2. Scaricare la **Immagine Docker** di FlashSyn

```
lm@lucam:~/project_FlashSyn$ sudo docker pull zhiychen597/flashsyn:latest
latest: Pulling from zhiychen597/flashsyn
16ea0e8c8879: Pull complete
50024b0106d5: Pull complete
ff95660c6937: Pull complete
9c7d0e5c0bc2: Pull complete
29c4fb388fdf: Pull complete
8659dae93050: Pull complete
1da0ab556051: Pull complete
e92ae9350d4a: Pull complete
c648cb7fc575: Pull complete
ea3ee54b8ae5: Pull complete
821dd0780458: Pull complete
83f83b9c3793: Pull complete
3f4d44b79139: Pull complete
c26215707004: Pull complete
d65745795ba1: Pull complete
9bd0f1e6ed4e: Pull complete
a1876a2ebb87: Pull complete
bbb9fc2d3312: Pull complete
Digest: sha256:21fca8cdef66d092825239f39fa4b6f193763e71b70139cfdccd7d1232a961c8
Status: Downloaded newer image for zhiychen597/flashsyn:latest
docker.io/zhiychen597/flashsyn:latest
lm@lucam:~/project_FlashSyn$
```

# Steps per eseguire FlashSyn in Docker (2/2)

3. Creazione di un **Docker volume**: serve a non perdere i log se il container si chiude

```
lm@lucam:~/project_FlashSyn$ sudo docker volume create FlashSyn-Data-Reproduce  
FlashSyn-Data-Reproduce  
lm@lucam:~/project_FlashSyn$
```

4. Avviare **Docker container**

```
lm@lucam:~/project_FlashSyn$ sudo docker run -it -v FlashSyn-Data-Reproduce:/FlashSyn/Results-To-Reproduce/ zhiychen597/  
flashsyn:latest bash  
root@42b03ffe505b:/FlashSyn#
```

5. Preparare l'**ambiente** all'interno del container

```
root@42b03ffe505b:/FlashSyn# . ~/.bashrc  
root@42b03ffe505b:/FlashSyn# forge -V  
forge 0.2.0 (6fc06c5 2024-01-05T02:43:33.315449279Z)
```

# Test di Base

## 1. Esecuzione del test di base di FlashSyn

```
root@42b03ffe505b:/FlashSyn# ls
Benchmarks          LICENSE      README3.md        flashsyn.tar      runRQ1.sh  runRQ4.sh  src
Dockerfile          README.md   Results-Expected  paper            runRQ2.sh  runTest.sh  temp.txt
HOW-TO-READ-DATA-LOG.md README2.md Results-To-Reproduce requirements.txt  runRQ3.sh  settings.toml
root@42b03ffe505b:/FlashSyn# chmod +x ./runTest.sh && ./runTest.sh
===== Current Date and Time: Wed Apr 30 08:15:45 UTC 2025 =====
running FlashSyn-precise baseline for bEarnFi...
bEarnFi (precise) done.
=====
===== ALL COMPLETE =====
=====
root@42b03ffe505b:/FlashSyn#
```

## 2. Verifica dei risultati del test

```
=====
===== End of Synthesis, time in total: 26.944101333618164 s =====
=====
===== Now shows the answers: =====
===== End of Answers =====
===== Best Profit 19.200000000000003 =====
```

**N.B.** il *best profit* riprodotto è diverso dal *profitto atteso* (*expected profit*).



# Riproduzione degli esperimenti

Il lavoro di FlashSyn si basa su quattro domande:

- **RQ1:** *Quanto è efficace FlashSyn nel sintetizzare vettori di attacco flash loan?*
- **RQ2:** *Quanto bene funziona l'approccio synthesis-via-approximation rispetto a sintetizzatori precisi?*
- **RQ3:** *Quanto migliora il risultato di FlashSyn se si utilizza un approccio counterexample-driven refinement?*
- **RQ4:** *Quanto è efficace l'integrazione tra FlashFind (per trovare azioni) e FlashSyn (per creare attacchi)?*

## Note importanti:

1. *shgo solver è non deterministico* → usa **strategie diverse a seconda dell'hardware** (ciò che conta è trovare un attacco con profitto positivo, non il valore esatto).
2. Gli autori usano fino a **18 processi paralleli**, ma nella versione Docker fornita è limitato a 1 processore → FlashSyn potrebbe essere più lento e trovare risultati leggermente meno buoni.
3. L'immagine Docker è stata testata solo su **Ubuntu AMD**. Potrebbero esserci problemi su **macOS** o **Windows**, soprattutto ARM.

# RQ1 – Setup per l'esecuzione

- È stata svolta l'esecuzione degli esperimenti su *tre Benchmark*: **bEarnFi**, **Wdoge** e **CheeseBank**. Per ognuno di essi, l'esecuzione ha previsto:
  - **2000 initial data points**
  - **Counterexample-driven refinement**
  - Utilizzo di **polynomial regression (FlashSyn-poly)** e **nearest-neighbor interpolation (FlashSyn-inte)** → tecniche approssimate.

Benchmark	FlashSyn-poly								FlashSyn-inter			Precise
	AC	AP	GL	GP	IDP	TDP	Profit	Time	TDP	Profit	Time	Profit
bZx1	3	3	2	1194	5192	5849	2392	422	6373	2302 <sup>†</sup>	441	cs
Harvest_USDT	4	4	4	338448	8000	9325	110139 <sup>†</sup>	670	10289	86798 <sup>†</sup>	7579	cs
Harvest_USDC	4	4	4	307416	8000	8912	59614 <sup>†</sup>	677	10914	110051 <sup>†</sup>	8349	cs
Eminence	4	4	5	1674278	8000	8780	1507174	1191	8104	/	/	1606965
ValueDeFi	6	6	6	8618002	12000	19975	8378194 <sup>†</sup>	4691	15758	6428341 <sup>†</sup>	11089	cx
CheeseBank	8	3	8	3270347	2679	2937	1946291 <sup>†</sup>	4391	2715	1101547 <sup>†</sup>	10942	2816762 <sup>†</sup>
Warp	6	3	6	1693523	6000	6000	2773345 <sup>†</sup>	1164	6000	/	/	2645640 <sup>†</sup>
bEarnFi	2	2	4	18077	4000	4854	13770	470	4652	12329	688	13832
AutoShark	8	3	8	1381	2753	2753	1372 <sup>†</sup>	5484	2753	/	/	cx
ElevenFi	5	2	5	129741	4000	4070	129658	409	4326	85811	898	cx
ApeRocket	7	3	6	1345	6000	6402	1333 <sup>†</sup>	733	6235	1037 <sup>†</sup>	3238	cs
Wdoge	5	1	5	78	2000	2001	75	272	2080	75	289	75
Novo	4	2	4	24857	4000	4164	20210	702	4031	23084	861	cx
OneRing	2	2	2	1534752	4000	4710	1814882	585	4218	1942188	367	cx
Puppet	3	3	2	89000	6000	6301	89000 <sup>†</sup>	1203	6452	87266 <sup>†</sup>	1238	89000 <sup>†</sup>
PuppetV2	4	3	3	953100	4491	4836	747799 <sup>†</sup>	2441	5061	362541 <sup>†</sup>	2835	647894 <sup>†</sup>
						Solved:16/18 Avg. Time: 1594			Solved:13/18 Avg. Time: 3754			

# RQ1 – Analisi tabella

- **AC (Action Candidates)**: numero totale di azioni candidate disponibili nel protocollo (es. deposit, swap, withdraw).
- **AP (Actions to Approximate)**: quante di queste azioni richiedono approssimazioni (perché troppo complesse).
- **GL (Ground Truth Length)**: lunghezza (in numero di azioni) del vettore d'attacco reale (storico) che è realmente avvenuto sulla blockchain.
- **GP (Ground Truth Profit)**: profitto ottenuto dall'attacco storico.
- **IDP (Initial Data Points)**: numero di data points raccolti all'inizio da FlashSyn per stimare le funzioni.
- **TDP (Total Data Points)**: numero di punti totali usati (dopo refinement con controesempi).

Benchmark	FlashSyn-poly								FlashSyn-inter			Precise
	AC	AP	GL	GP	IDP	TDP	Profit	Time	TDP	Profit	Time	Profit
bZx1	3	3	2	1194	5192	5849	2392	422	6373	2302 <sup>†</sup>	441	cs
Harvest_USDT	4	4	4	338448	8000	9325	110139 <sup>†</sup>	670	10289	86798 <sup>†</sup>	7579	cs
Harvest_USDC	4	4	4	307416	8000	8912	59614 <sup>†</sup>	677	10914	110051 <sup>†</sup>	8349	cs
Eminence	4	4	5	1674278	8000	8780	1507174	1191	8104	/	/	1606965
ValueDeFi	6	6	6	8618002	12000	19975	8378194 <sup>†</sup>	4691	15758	6428341 <sup>†</sup>	11089	cx
CheeseBank	8	3	8	3270347	2679	2937	1946291 <sup>†</sup>	4391	2715	1101547 <sup>†</sup>	10942	2816762 <sup>†</sup>
Warp	6	3	6	1693523	6000	6000	2773345 <sup>†</sup>	1164	6000	/	/	2645640 <sup>†</sup>
bEarnFi	2	2	4	18077	4000	4854	13770	470	4652	12329	688	13832
AutoShark	8	3	8	1381	2753	2753	1372 <sup>†</sup>	5484	2753	/	/	cx
ElevenFi	5	2	5	129741	4000	4070	129658	409	4326	85811	898	cx
ApeRocket	7	3	6	1345	6000	6402	1333 <sup>†</sup>	733	6235	1037 <sup>†</sup>	3238	cs
Wdoge	5	1	5	78	2000	2001	75	272	2080	75	289	75
Novo	4	2	4	24857	4000	4164	20210	702	4031	23084	861	cx
OneRing	2	2	2	1534752	4000	4710	1814882	585	4218	1942188	367	cx
Puppet	3	3	2	89000	6000	6301	89000 <sup>†</sup>	1203	6452	87266 <sup>†</sup>	1238	89000 <sup>†</sup>
PuppetV2	4	3	3	953100	4491	4836	747799 <sup>†</sup>	2441	5061	362541 <sup>†</sup>	2835	647894 <sup>†</sup>
						Solved:16/18 Avg. Time: 1594			Solved:13/18 Avg. Time: 3754			

**N.B.** i **data points** rappresentano **coppie input-output** (*prestate - poststate*) raccolte durante l'esecuzione di specifiche azioni (*funzioni dei contratti DeFi*) su una blockchain privata (*forkata*). Questi dati servono a costruire un modello approssimato del comportamento di tali azioni.

# RQ1 – Esecuzione bEarnFi

- Si esegue FlashSyn su di un solo benchmark usando entrambe le *tecniche approssimate*:

```
root@ec0ef2b40145:/FlashSyn# ./runRQ1.sh bEarnFi
===== Current Date and Time: Thu May  1 10:08:55 UTC 2025 =====
running FlashSyn-inter with 2000 initial datapoints with counterexample driven loops for bEarnFi...
bEarnFi done.
Benchmarks: 1/32 finished, 31/32 to do
===== Current Date and Time: Thu May  1 10:16:50 UTC 2025 =====
running FlashSyn-poly with 2000 initial datapoints with counterexample driven loops for bEarnFi...
bEarnFi done.
Benchmarks: 2/32 finished, 30/32 to do
=====
===== ALL COMPLETE =====
=====
```

**N.B.** 32 esecuzioni (16 benchmarks x 2 metodi).

# RQ1 – Analisi dei risultati di bEarnFi (1/2)

- Si esegue lo *script python RQ1.py* per analizzare l'output:

```
root@ec0ef2b40145:/FlashSyn# python3 Results-To-Reproduce/RQ1_test1.py
|| FlashSyn-poly || FlashSyn-inte ||
benchmark GP IDP TDP Profit Time TDP Profit Time
bEarnFi 18077 4000 4000 / / 4000 / /
Wdoge 78 2000 2000 / / 2000 / /
Profit_inte [0, 0]
len: 2
Profit_poly [0, 0]
len: 2
Profit_his [18077, 78]
len: 2
Time_inte []
len: 0
Time_poly ['/', '/']
len: 2
=====
Solved(poly): 0 out of 18 Avg Time: / Solved(inte): 0 out of 18 Avg Time: /
root@ec0ef2b40145:/FlashSyn#
```

**N.B.** è necessario modificare lo script (**RQ1\_test1.py**).

**N.B.** Apparentemente, sembra che non siano stati trovati attacchi. Questo accade perché la soglia di allarme è settata a 40 (*profit* > 40).

# RQ1 – Analisi dei risultati di bEarnFi (2/2)

- bEarnFi\_inte.txt:

```
Deposit number of points:
2000
EmergencyWithdraw number of points:
2000
=====
===== End of Synthesis, time in total: 460.6286678314209 s =====
=====
===== Now shows the answers: =====
===== End of Answers =====
===== Best Profit 16.0 =====
```

- bEarnFi\_poly.txt:

```
Deposit number of points:
2000
EmergencyWithdraw number of points:
2000
=====
===== End of Synthesis, time in total: 50.22027611732483 s =====
=====
===== Now shows the answers: =====
===== End of Answers =====
===== Best Profit 5.6000000000000005 =====
```

# RQ1 – Esecuzione Wdodge

- Si esegue FlashSyn su di un solo benchmark usando entrambe le *tecniche approssimate*:

```
root@ec0ef2b40145:/FlashSyn# ./runRQ1.sh Wdodge
===== Current Date and Time: Thu May  1 10:25:54 UTC 2025 =====
running FlashSyn-inter with 2000 initial datapoints with counterexample driven loops for Wdodge...
Wdodge done.
Benchmarks: 1/32 finished, 31/32 to do
===== Current Date and Time: Thu May  1 10:27:25 UTC 2025 =====
running FlashSyn-poly with 2000 initial datapoints with counterexample driven loops for Wdodge...
Wdodge done.
Benchmarks: 2/32 finished, 30/32 to do
=====
===== ALL COMPLETE =====
=====
```

**Note:** 32 esecuzioni (16 benchmarks x 2 metodi).

# RQ1 – Analisi dei risultati di Wdoge (1/2)

- Si esegue lo *script python RQ1.py* per analizzare l'output:

```
root@ec0ef2b40145:/FlashSyn# python3 Results-To-Reproduce/RQ1_test1.py
|| FlashSyn-poly || FlashSyn-inte ||
benchmark GP IDP TDP Profit Time TDP Profit Time
bEarnFi 18077 4000 4000 / / 4000 / /
Wdoge 78 2000 2000 / / 2000 / /
Profit_inte [0, 0]
len: 2
Profit_poly [0, 0]
len: 2
Profit_his [18077, 78]
len: 2
Time_inte []
len: 0
Time_poly ['/', '/']
len: 2
=====
Solved(poly): 0 out of 18 Avg Time: / Solved(inte): 0 out of 18 Avg Time: /
root@ec0ef2b40145:/FlashSyn#
```

**N.B.** è necessario modificare lo script (**RQ1\_test1.py**).

**N.B.** Apparentemente, sembra che non siano stati trovati attacchi. Questo accade perché la soglia di allarme è settata a 40 ( $profit > 40$ ).



# RQ1 – Analisi dei risultati di Wdoge (2/2)

- Wdoge\_inte.txt:

```
SwapWBNB2Wdoge number of points:
skip
TransferWdoge number of points:
skip
PancakePairSkim number of points:
2000
PancakePairSync2 number of points:
skip
SwapWdoge2WBNB number of points:
skip
=====
===== End of Synthesis, time in total:  89.1891930103302 s =====
=====
===== Now shows the answers: =====
===== End of Answers =====
===== Best Profit  5.0 =====
```

- Wdoge\_poly.txt:

```
SwapWBNB2Wdoge number of points:
skip
TransferWdoge number of points:
skip
PancakePairSkim number of points:
2000
PancakePairSync2 number of points:
skip
SwapWdoge2WBNB number of points:
skip
=====
===== End of Synthesis, time in total:  30.672540426254272 s =====
=====
===== Now shows the answers: =====
===== End of Answers =====
===== Best Profit  4.2 =====
```

# RQ1 – Esecuzione CheeseBank

- Si esegue FlashSyn su di un solo benchmark usando entrambe le *tecniche approssimate*:

```
root@ebfbe281b202:/FlashSyn# ./runRQ1.sh CheeseBank
===== Current Date and Time: Sat May  3 10:34:32 UTC 2025 =====
running FlashSyn-inter with 2000 initial datapoints with counterexample driven loops for CheeseBank...
./runRQ1.sh: line 28: 12 Killed                  timeout 10800s python3.7 src/main.py $case 0 1 2000 > ./${dir}/${case}
_inte.txt
CheeseBank done.
Benchmarks: 1/32 finished, 31/32 to do
===== Current Date and Time: Sat May  3 13:09:26 UTC 2025 =====
running FlashSyn-poly with 2000 initial datapoints with counterexample driven loops for CheeseBank...
CheeseBank done.
Benchmarks: 2/32 finished, 30/32 to do
=====
===== ALL COMPLETE =====
=====
root@ebfbe281b202:/FlashSyn#
```

**Note:** 32 esecuzioni (16 benchmarks x 2 metodi).

# RQ1 – Analisi dei risultati di CheeseBank (1/4)

- Si esegue lo script *python RQ1.py* per analizzare l'output:

```
root@ebfbe281b202:/FlashSyn# python3 Results-To-Reproduce/RQ1_CheeseBank.py
|| FlashSyn-poly || FlashSyn-inte ||
benchmark GP TDP TDP Profit Time TDP Profit Time
CheeseBank 3270347 2679 2679 / / 2679 / /
Profit_inte [0]
len: 1
Profit_poly [0]
len: 1
Profit_his [3270347]
len: 1
Time_inte []
len: 0
Time_poly ['/']
len: 1
=====
Solved(poly): 0 out of 18 Avg Time: / Solved(inte): 0 out of 18 Avg Time: /
root@ebfbe281b202:/FlashSyn#
```

**N.B.** è necessario modificare lo script (**RQ1\_CheeseBank.py**).

**N.B.** Apparentemente, sembra che non siano stati trovati attacchi. Questo accade perché la soglia di allarme è settata a 40 (*profit* > 40).

# RQ1 – Analisi dei risultati di CheeseBank (2/4)

- CheeseBank\_inte.txt: è composto da differenti sezioni

```
===== Strength: 0 Last Profit: 0 =====
Now global best profit is, 0.6000000000000001
For Symbolic Attack Vector: SwapUniswapETH2LP, LP2LQ, SwapUniswapETH2Cheese, SwapUniswapCheese2ETH, RefreshCheeseBank,
BorrowCheese_USDC, BorrowCheese_USDT, BorrowCheese_DAI
[1, 1, 1, 3322, 2215, 1326, 94]
Estimated Profit -890.9266567456768 Actual Profit 0
===== Best Profit: 0.0 Best Paras: [], time: 1591.2785892486572
===== Strength: 0 Last Profit: 0 =====
Now global best profit is, 0.6000000000000001
```

```
Now global best profit is, 0.6000000000000001
===== in total 2278 concrete attack vectors are checked =====
===== in total 0 executions succeed
===== Next round we have 775 symbolic attack vectors to check:
===== Pruning 1 choose the best 100 trace candidates
===== Next round we have 100 symbolic attack vectors to check:
=====
===== Strength 0 - round 0 of concrete attack vector verification finishes =====
===== Best Global Profit: 0.6000000000000001 =====
=====
SwapUniswapETH2LP number of points:
skip
SwapUniswapETH2Cheese number of points:
skip
RefreshCheeseBank number of points:
2000
LP2LQ number of points:
567
BorrowCheese_USDC number of points:
skip
BorrowCheese_USDT number of points:
skip
BorrowCheese_DAI number of points:
112
SwapUniswapCheese2ETH number of points:
skip
```

# RQ1 – Analisi dei risultati di CheeseBank (3/4)

- **Estimated profits** con soluzioni ottimali relative trovate dall'ottimizzatore:

```
Check Contract:      SwapUniswapETH2Cheese, SwapUniswapCheese2ETH   Profit of Previous Iteration:  0.60000000000000
01 time: 1591.3402297496796
The optimizer takes 0.013136148452758789 seconds
best para: [1.e+00 3.e+05] best profit: 18956.58813860811
Optimization terminated successfully.  Next only show the first 5/5 profitable solutions
[1, 1]      estimated profit is, -443.0683038601477
[1, 24172]  estimated profit is, 2476.8626445061295
[1, 300000] estimated profit is, 18956.58813860811
[7, 295057] estimated profit is, 18130.34057407173
[657, 159375] estimated profit is, -19635.193317938396

Check Contract:      RefreshCheeseBank, SwapUniswapETH2Cheese, RefreshCheeseBank, SwapUniswapETH2Cheese, SwapUniswapC
heese2ETH   Profit of Previous Iteration:  0.4 time: 1591.377363204956
The optimizer takes 0.05618739128112793 seconds
best para: [1.e+00 1.e+00 3.e+05] best profit: 18850.0278246223
Optimization terminated successfully.  Next only show the first 2/2 profitable solutions
[1, 1, 300000] estimated profit is, 18850.0278246223
[2, 3, 292456] estimated profit is, 18162.06157726071
[1, 1, 299400] estimated profit is, 18830.739722751696
[1, 2, 291871] estimated profit is, 18477.00188531007
```

- Nessuno degli attacchi generati produce un profitto quando viene eseguito con **Foundry**:

```
forge test --match-contract attackTester --fork-url https://rpc.ankr.com/eth/d81f3fbb1f894af172b06e04687b43b7d94d335c233
1656722ede40d9888a46e --fork-block-number 11205646
b'Compiling 2 files with 0.7.6\nSolc 0.7.6 finished in 5.77s\nCompiler run successful (with warnings)\nwarning[5574]: sr
c/attack.sol:18:1: Warning: Co
Running attacks on foundry costs time: 16.143885374069214 seconds
For Symbolic Attack Vector: SwapUniswapETH2Cheese, SwapUniswapCheese2ETH
[1, 1]
Estimated Profit -443.0683038601477 Actual Profit 0
[1, 24172]
Estimated Profit 2476.8626445061295 Actual Profit 0
[1, 300000]
Estimated Profit 18956.58813860811 Actual Profit 0
[7, 295057]
Estimated Profit 18130.34057407173 Actual Profit 0
[657, 159375]
Estimated Profit -19635.193317938396 Actual Profit 0
[1, 24123]
Estimated Profit 2489.2925499547914 Actual Profit 0
[1, 299400]
Estimated Profit 18937.517269459902 Actual Profit 0
[6, 294466]
Estimated Profit 18248.61381042906 Actual Profit 0
===== Best Profit: 0.6000000000000001 Best Paras: [], time: 1680.664190530777
===== Strength: 0 Last Profit: 0.6000000000000001 =====
Now global best profit is, 0.6000000000000001
```

# RQ1 – Analisi dei risultati di CheeseBank (4/4)

- Ultime righe:

```
Check Contract: RefreshCheeseBank, SwapUniswapETH2LP, LP2LQ, SwapUniswapETH2Cheese, BorrowCheese_DAI, BorrowCheese_USDC, SwapUniswapCheese2ETH, BorrowCheese_USDT Profit of Previous Iteration: 0.4 time: 8145.3838312625885
The optimizer takes 58.74723696708679 seconds
best para: [1.00000000e+00 5.75523210e+01 1.00000000e+00 8.75860000e+04
3.68690175e+05 3.00000000e+05 2.12881688e+04] best profit: 474793.0119027301
Optimization terminated successfully. Next only show the first 2/2 profitable solutions
[1, 57, 1, 87586, 368690, 300000, 21288] estimated profit is, 474793.0119027301
[29, 901, 15432, 21255, 178751, 270263, 180743] estimated profit is, 344408.01487132703
[1, 56, 1, 87410, 367952, 299400, 21245] estimated profit is, 473859.32040719455
[28, 899, 15401, 21212, 178393, 269722, 180381] estimated profit is, 345971.4179935864

Check Contract: RefreshCheeseBank, SwapUniswapETH2LP, LP2LQ, SwapUniswapETH2Cheese, BorrowCheese_DAI, BorrowCheese_USDC, BorrowCheese_USDT, SwapUniswapCheese2ETH Profit of Previous Iteration: 0.4 time: 8204.37356185913
The optimizer takes 60.078553199768066 seconds
best para: [1.00000000e+00 7.24063891e+01 1.00000000e+00 8.75860000e+04
3.20859278e+05 5.15996821e+04 3.00000000e+05] best profit: 426962.1145129993
Optimization terminated successfully. Next only show the first 2/2 profitable solutions
[1, 72, 1, 87586, 320859, 51599, 300000] estimated profit is, 426962.1145129993
[87, 509, 15863, 83779, 148454, 122712, 186474] estimated profit is, 157221.75618578758
[1, 71, 1, 87410, 320217, 51495, 299400] estimated profit is, 477619.32040719455
[86, 507, 15831, 83611, 148157, 122466, 186101] estimated profit is, 159462.1302208551

root@ebf8e281b202:/FlashSyn#
```

- **N.B.** i risultati ottenuti sono consistenti con quelli dello script python ma non con quelli attesi del paper (tabella 3).

# RQ1 – Conclusioni

- **Efficacia complessiva:** FlashSyn è stato in grado di individuare attacchi con profitto positivo per alcuni benchmark (bEarnFi, Wdoge) ma non per altri (CheeseBank).
- **Tecniche approssimative:** le tecniche di interpolazione polinomiale e regressione funzionano, ma non sempre individuano attacchi validi o concreti.

**N.B.** stime di profitto elevate ma nessuna esecuzione valida (nessun attacco concreto produce profitto). I profitti sono spesso molto inferiori a quelli riportati nella tabella del documento (ad esempio, 0,6 contro 2,8 milioni per CheeseBank).



# RQ2 – Esecuzione bEarnFi

- Esecuzione dello **script bash**:

```
root@ec0ef2b40145:/FlashSyn# chmod +x ./runRQ2.sh
root@ec0ef2b40145:/FlashSyn# ./runRQ2.sh bEarnFi
===== Current Date and Time: Thu May 1 16:34:45 UTC 2025 =====
running FlashSyn-precise baseline for bEarnFi...
bEarnFi (precise) done
Benchmarks: 1/7 finished, 6/7 to do
=====
===== ALL COMPLETE =====
=====
```

- **N.B.** lo script bash distingue sette benchmark e tre categorie (*easy*, *mid* e *hard*).



```
# Check for user input
if [ $# -eq 0 ]; then
    # Easy:
    for case in 'bEarnFi' 'Puppet' 'PuppetV2' ; do
        run_case "$case"
    done

    # Mid:
    for case in 'Eminence' 'Wdoge'; do
        run_case "$case"
    done

    # Hard:
    for case in 'CheeseBank' 'Warp'; do
        run_case "$case"
    done
else
    # Argument provided, run for the specified case
    run_case "$1"
fi
```



# RQ2 – Analisi dei risultati di bEarnFi (1/2)

- Esecuzione dello *script python* (**RQ2.py**) per analizzare l'output:

```
root@ec0ef2b40145:/FlashSyn# python3 Results-To-Reproduce/RQ2_test1.py  
benchmarkprecise  
bEarnFi /  
Wdoge /  
root@ec0ef2b40145:/FlashSyn#
```

**N.B.** Dobbiamo modificare alcuni script (**RQ2\_test1.py**).

**N.B.** Sono stati eseguiti bEarnFi e Wdoge nello stesso istante.

# RQ2 – Analisi dei risultati di bEarnFi (2/2)

- bEarnFi\_precise.txt (*riprodotto*):

```
Deposit number of points:
skip
EmergencyWithdraw number of points:
0
=====
===== End of Synthesis, time in total: 57.75315546989441 s =====
=====
===== Now shows the answers: =====
===== End of Answers =====
===== Best Profit 19.200000000000003 =====
```

- Check con **grep**:

```
root@ec0ef2b40145:/FlashSyn# grep "Best Global Profit:" Results-To-Reproduce/FlashSynData/precise/bEarnFi_precise.txt
===== Best Global Profit: 0.2 =====
===== Best Global Profit: 0.2 =====
===== Best Global Profit: 5.0 =====
===== Best Global Profit: 5.0 =====
===== Best Global Profit: 19.200000000000003 =====
===== Best Global Profit: 19.200000000000003 =====
root@ec0ef2b40145:/FlashSyn#
```

# RQ2 – Confronto risultati bEarnFi

- bEarnFi\_precise.txt (*expected*):

```
Deposit number of points:
skip
EmergencyWithdraw number of points:
0
=====
===== End of Synthesis, time in total: 110.64523196220398 s =====
=====
===== Now shows the answers: =====
For Symbolic Attack Vector: Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw
Best Profit: 13832 Parameters: [7804238, 7800912]
===== End of Answers =====
===== Best Profit 13832 =====
root@ec0ef2b40145:/FlashSyn/Results-Expected/FlashSynData/precise#
```

Aspect	Reproduce Results	Expected Results	Difference
Synthesis Time	~57.75 s	~110.64 s	First is faster (less exploration?)
Valid Data Points	skip / 0	skip / 0	-
Attack Vector	Not shown	Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw	First output is truncated or simplified
Attack Parameters	Not shown	[78004238, 7800912]	Not available in first output
Best Profit	19.2	13832	Much lower in first case

# RQ2 – Esecuzione Wdoge

- Esecuzione dello **script bash**:

```
root@ec0ef2b40145:/FlashSyn# ./runRQ2.sh Wdoge
===== Current Date and Time: Thu May 1 16:36:29 UTC 2025 =====
running FlashSyn-precise baseline for Wdoge...
Wdoge (precise) done
Benchmarks 1/7 finished, 6/7 to do
=====
===== ALL COMPLETE =====
=====
root@ec0ef2b40145:/FlashSyn#
```

- **N.B.** lo script bash distingue sette benchmark e tre categorie (*easy*, *mid* e *hard*).



```
# Check for user input
if [ $# -eq 0 ]; then
    # Easy:
    for case in 'bEarnFi' 'Puppet' 'PuppetV2' ; do
        run_case "$case"
    done

    # Mid:
    for case in 'Eminence' 'Wdoge'; do
        run_case "$case"
    done

    # Hard:
    for case in 'CheeseBank' 'Warp'; do
        run_case "$case"
    done
else
    # Argument provided, run for the specified case
    run_case "$1"
fi
```

# RQ2 – Analisi dei risultati di Wdoge (1/2)

- Esecuzione dello *script python* (**RQ2.py**) per analizzare l'output:

```
root@ec0ef2b40145:/FlashSyn# python3 Results-To-Reproduce/RQ2_test1.py  
benchmarkprecise  
bEarnFi /  
Wdoge /  
root@ec0ef2b40145:/FlashSyn#
```

**N.B.** Dobbiamo modificare alcuni script (**RQ2\_test1.py**).

**N.B.** Sono stati eseguiti bEarnFi e Wdoge nello stesso istante.

# RQ2 – Analisi dei risultati di Wdoge (2/2)

- Wdoge\_precise.txt (riprodotto):

```
SwapWBNB2Wdoge number of points:
skip
TransferWdoge number of points:
skip
PancakePairSkim number of points:
skip
PancakePairSync2 number of points:
skip
SwapWdoge2WBNB number of points:
skip
=====
===== End of Synthesis, time in total: 72.46661829948425 s =====
=====
===== Now shows the answers: =====
===== End of Answers =====
===== Best Profit 4.2 =====
```

- Check with grep:

```
root@ec0ef2b40145:/FlashSyn# grep "Best Global Profit:" Results-To-Reproduce/FlashSynData/precise/Wdoge_precise.txt
===== Best Global Profit: 0.4 =====
===== Best Global Profit: 0.4 =====
===== Best Global Profit: 1.2000000000000002 =====
===== Best Global Profit: 1.2000000000000002 =====
===== Best Global Profit: 4.2 =====
===== Best Global Profit: 4.2 =====
root@ec0ef2b40145:/FlashSyn#
```

# RQ2 – Confronto risultati Wdoge

- Wdoge\_precise.txt (*expected*):

```
SwapWBNB2Wdoge number of points:
skip
TransferWdoge number of points:
skip
PancakePairSkim number of points:
skip
PancakePairSync2 number of points:
skip
SwapWdoge2WBNB number of points:
skip
=====
===== End of Synthesis, time in total: 86.55322694778442 s =====
=====
===== Now shows the answers: =====
For Symbolic Attack Vector: SwapWBNB2Wdoge, TransferWdoge, PancakePairSkim, PancakePairSync2, SwapWdoge2WBNB
Best Profit: 75 Parameters: [2859, 5156250, 4609375]
===== End of Answers =====
===== Best Profit 75 =====
root@ec0ef2b40145:/FlashSyn/Results-Expected/FlashSynData/precise#
```

Aspect	Reproduce Results	Expected Results	Difference
Synthesis Time	~72.47 s	~86.55 s	Similar, so time is not the issue
Valid Data Points	skip / skip	skip / skip	Equal
Attack Vector	Not shown	SwapWBNB2Wdoge, TransferWdoge, PancakePairSkim, PancakePairSync2, SwapWdoge2WBNB	First output is truncated or simplified
Attack Parameters	Not shown	[2859, 5156250, 4609375]	Not available in first output
Best Profit	4.2	75	Much lower in first case

# RQ2 – Esecuzione CheeseBank

- Esecuzione dello **script bash**:

```
root@ebfbe281b202:/FlashSyn# chmod +x ./runRQ2.sh
root@ebfbe281b202:/FlashSyn# ./runRQ2.sh CheeseBank
===== Current Date and Time: Mon May 5 08:31:56 UTC 2025 =====
running FlashSyn-precise baseline for CheeseBank...
CheeseBank (precise) done
Benchmarks: 1/7 finished, 6/7 to do
=====
===== ALL COMPLETE =====
=====
root@ebfbe281b202:/FlashSyn#
```

- **N.B.** lo script bash distingue sette benchmark e tre categorie (*easy*, *mid* e *hard*).



```
# Check for user input
if [ $# -eq 0 ]; then
    # Easy:
    for case in 'bEarnFi' 'Puppet' 'PuppetV2' ; do
        run_case "$case"
    done

    # Mid:
    for case in 'Eminence' 'Wdoge'; do
        run_case "$case"
    done

    # Hard:
    for case in 'CheeseBank' 'Warp'; do
        run_case "$case"
    done
else
    # Argument provided, run for the specified case
    run_case "$1"
fi
```



# RQ2 – Analisi dei risultati di CheeseBank (1/2)

- Esecuzione dello *script python* (**RQ2.py**) per analizzare l'output:

```
root@ebfbe281b202:/FlashSyn# python3 Results-To-Reproduce/RQ2_CheeseBank.py  
benchmarkprecise  
CheeseBank New global Profit: 0.6000000000000001  
/  
root@ebfbe281b202:/FlashSyn#
```

**N.B.** Dobbiamo modificare alcuni script (**RQ2\_test1.py**).

**N.B.** il **/ finale** rappresenta l'assenza di un Actual Profit > 0 oppure di un valid runtime.

# RQ2 – Analisi dei risultati di CheeseBank (2/2)

- CheeseBank\_precise.txt (riprodotto):

```
Check Contract: RefreshCheeseBank, SwapUniswapETH2LP, SwapUniswapETH2Cheese, LP2LQ, BorrowCheese_DAI, BorrowCheese_USDC, BorrowCheese_USDT, SwapUniswapCheese2ETH Profit of Previous Iteration: 0.2 time: 7621.208479166031
The optimizer takes 112.23292708396912 seconds
best para: [1.00000000e+00 1.00000000e+00 7.84336677e+00 8.75860000e+04
8.09548591e+02 2.16798619e+02 3.00000000e+05] best profit: 18516.836811552836
Optimization terminated successfully. Next only show the first 1/1 profitable solutions
[1, 1, 7, 87586, 809, 216, 300000] estimated profit is, 18516.836811552836
[1, 1, 6, 87410, 807, 215, 299400] estimated profit is, 18497.320407194526

Check Contract: SwapUniswapETH2LP, LP2LQ, BorrowCheese_DAI, BorrowCheese_USDT, RefreshCheeseBank, SwapUniswapETH2Cheese, SwapUniswapCheese2ETH, BorrowCheese_USDC Profit of Previous Iteration: 0.2 time: 7735.72798538208
The optimizer takes 108.75004386901855 seconds
best para: [3.33256650e+01 5.72456088e+01 1.00000000e+00 1.00000000e+00
7.32739584e+02 2.21050653e+05 1.00000000e+00] best profit: -40212.90605325316
Optimization terminated successfully. Next only show the first 1/1 profitable solutions
[33, 57, 1, 1, 732, 221050, 1] estimated profit is, -40212.90605325316
```

- Check con grep:

```
root@ebfbe281b202:/FlashSyn# grep "Best Global Profit:" Results-To-Reproduce/FlashSynData/precise/CheeseBank_precise.txt
===== Best Global Profit: 0.6000000000000001 =====
===== Best Global Profit: 0.6000000000000001 =====
root@ebfbe281b202:/FlashSyn#
```

# RQ2 – Confronto risultati CheeseBank

- CheeseBank\_precise.txt (*expected*):

```
For Symbolic Attack Vector: SwapUniswapETH2LP, SwapUniswapETH2Cheese, RefreshCheeseBank, LP2LQ, BorrowCheese_USDC, SwapUn
iswapCheese2ETH, BorrowCheese_DAI
Best Profit: 1788587.8 Parameters: [73, 15281, 3338, 1997812, 121033, 54923]
For Symbolic Attack Vector: SwapUniswapETH2LP, SwapUniswapETH2Cheese, RefreshCheeseBank, SwapUniswapCheese2ETH, SwapUniswapETH2LP, LP2LQ, Bo
rrowCheese_USDT, BorrowCheese_DAI
Best Profit: 1144015.4 Parameters: [8995, 135535, 84, 2007, 1201272, 77033]
For Symbolic Attack Vector: SwapUniswapETH2LP, SwapUniswapETH2Cheese, RefreshCheeseBank, LP2LQ, BorrowCheese_USDC, Borrow
Cheese_DAI, SwapUniswapCheese2ETH, BorrowCheese_USDT
Best Profit: 2027910.8 Parameters: [67, 16620, 2827, 1692876, 21345, 259790, 409421]
For Symbolic Attack Vector: SwapUniswapETH2LP, SwapUniswapETH2Cheese, RefreshCheeseBank, LP2LQ, SwapUniswapCheese2ETH, Bo
rrowCheese_USDT, BorrowCheese_DAI, BorrowCheese_USDC
Best Profit: 1968274.8 Parameters: [89, 16399, 3974, 198091, 1193927, 39204, 923947]
For Symbolic Attack Vector: SwapUniswapETH2LP, SwapUniswapETH2Cheese, RefreshCheeseBank, LP2LQ, BorrowCheese_USDT, SwapUn
iswapCheese2ETH, BorrowCheese_DAI, BorrowCheese_USDC
Best Profit: 1130250.6 Parameters: [64, 13753, 2500, 424709, 102918, 13658, 967554]
For Symbolic Attack Vector: SwapUniswapETH2LP, SwapUniswapETH2Cheese, RefreshCheeseBank, LP2LQ, BorrowCheese_USDT, SwapUn
iswapCheese2ETH, BorrowCheese_USDC
Best Profit: 2043708.4 Parameters: [74, 18430, 3086, 986363, 47805, 1756715]
For Symbolic Attack Vector: SwapUniswapETH2LP, SwapUniswapETH2Cheese, RefreshCheeseBank, LP2LQ, BorrowCheese_USDT, Borrow
Cheese_USDC, BorrowCheese_DAI, SwapUniswapCheese2ETH
Best Profit: 2404532.6 Parameters: [99, 18221, 4033, 770724, 1986451, 33828, 115576]
For Symbolic Attack Vector: SwapUniswapETH2LP, SwapUniswapETH2Cheese, RefreshCheeseBank, LP2LQ, SwapUniswapCheese2ETH, Bo
rrowCheese_USDT, BorrowCheese_DAI
Best Profit: 1072649.0 Parameters: [33, 18782, 1263, 198651, 1054789, 86556]
For Symbolic Attack Vector: SwapUniswapETH2LP, SwapUniswapETH2Cheese, RefreshCheeseBank, LP2LQ, BorrowCheese_DAI, BorrowC
heese_USDT, SwapUniswapCheese2ETH, BorrowCheese_USDC
Best Profit: 2816762.4 Parameters: [83, 15091, 4084, 23587, 1005825, 277225, 1901696]
===== End of Answers =====
===== Best Profit 2816762.4 =====
root@ebfbc281b202:/FlashSyn/Results-Expected/FlashSynData/precise#
```

Aspect	Reproduce Results	Expected Results	Difference
Synthesis Time	~1680–1730 s	Not explicit (but successfully completed)	Relatively long time, but still complete
Valid Data Points	Some skip - e.g., BorrowCheese_USDC, USDT	Numerous valid points shown for each contract	Reproduce seems less exploratory or more conservative
Attack Vector	Shown in part (but with 0 profit results)	Explicit with details: SwapX, Refresh, Borrow...	More complete and detailed in expected results
Attack Parameters	Often negative or profit 0	Complete parameters, with positive profits	Reproductive identifies suboptimal strategies or ignores them
Best Profit	0.6000000000000001	2816762.4	Much lower in first case, probably for a failure during the execution

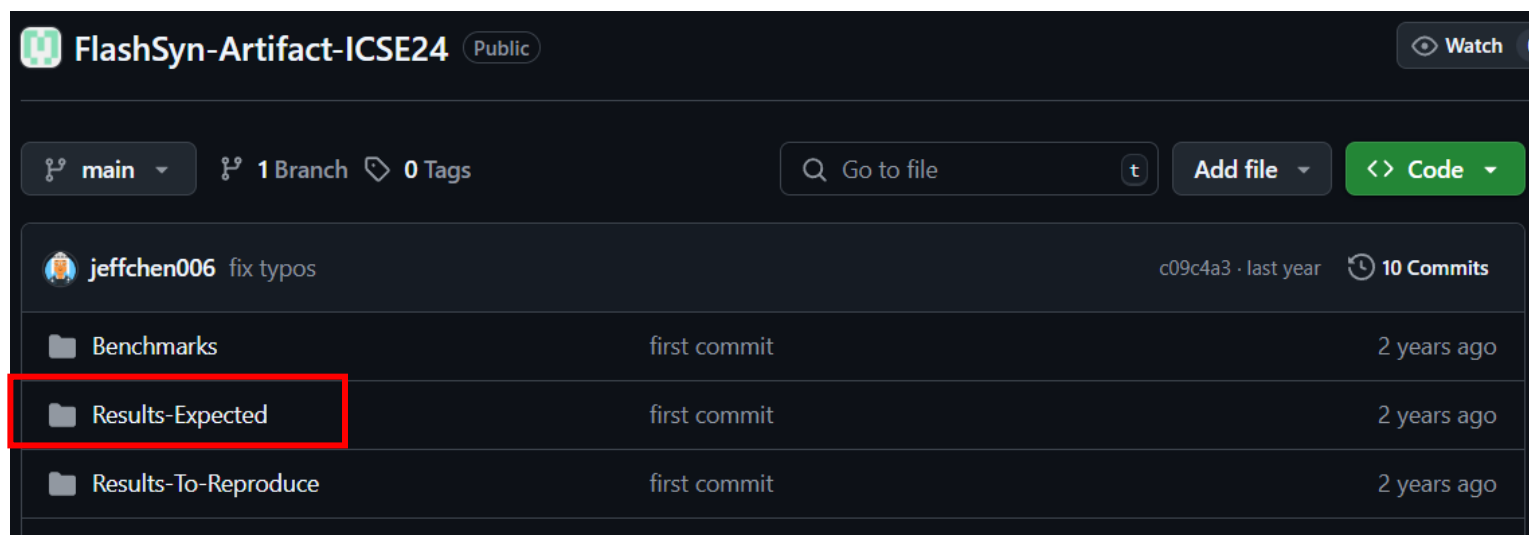
# RQ2 – Conclusioni

- **Tecnica precisa:** nei test precisi, i profitti stimati sono spesso elevati ma non vengono convalidati in modo concreto (profitto effettivo = 0).
- **Risultati incoerenti:** i file di log risultano strutturalmente diversi da quanto previsto, in particolare per CheeseBank.
- **Problemi con la generazione iniziale dei dati:** alcune funzioni vengono saltate (“skip”) → minore copertura → minori possibilità di individuare gli attacchi.

# Analisi CWE/CVE delle vulnerabilità DeFi

- **Secondo obiettivo:** prendere gli **attacchi generati da FlashSyn** contro protocolli DeFi e **classificarli** secondo *standard di sicurezza informatica*:
  - **CWE (Common Weakness Enumeration)** → *classificazione del tipo di vulnerabilità*
  - **CVE (Common Vulnerabilities and Exposures)** → *è un codice numerico che viene assegnato ad una vulnerabilità (schema di enumeration)*

**N.B.** L'**analisi** è stata fatta per i soli tre benchmark analizzati nei passi precedenti e, per questioni di consistenza con il paper fornito, sono stati analizzati i risultati (.txt) degli esperimenti svolti dagli autori dell'articolo (*Results-Expected*).



# Analisi dei file di output (1/2)

- Per prima cosa, è stato scritto uno **script python** che permettesse di ottenere un **file .csv** andando ad analizzare i tre benchmark presenti al seguente path: «*Results-Expected/FlashSynData/2000+X*», considerando per ciascuno di essi entrambe le metodologie (poly, inte).

- Nel file sono indicati:
  - **Protocollo**
  - **Tipo di analisi**
  - **Sequenze di funzioni contrattuali analizzate**
  - **Profitti stimati e reali**

**N.B.** Di fianco, è indicato un estratto del file .csv realizzato.

[illegible]



# Analisi dei file di output (2/2)

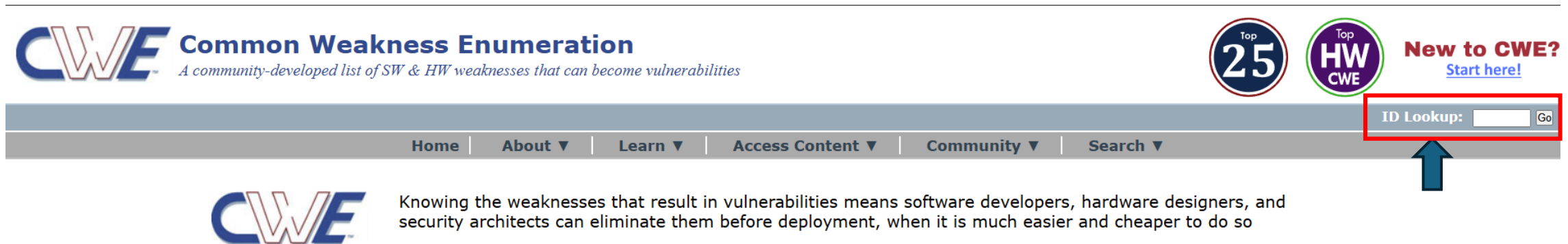
- Successivamente, è stato scritto un secondo **script python** che permettesse di ottenere un **file .csv** in cui venissero riportate due ulteriori colonne per ogni riga:
  - **CWE ID**
  - **CWE Description**

A1	Protocol,Type,Contract Sequence,Estimated Profit,Actual Profit,CWE ID,CWE Description																
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Protocol,Type,Contract Sequence,Estimated Profit,Actual Profit,CWE ID,CWE Description																
2	bEarnFi,inte,"Deposit, EmergencyWithdraw",4977.0,-1.0,Unclassified,Pattern not matched																
3	bEarnFi,inte,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw",5870.38812244311,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
4	bEarnFi,inte,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 8807",8219.624445812777,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
5	bEarnFi,inte,"Deposit, EmergencyWithdraw Profit of Previous Iteration: 3.9000000000000004",7914.441578073427,-1.0,Unclassified,Pattern not matched																
6	bEarnFi,inte,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 8807",9010.076699911617,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
7	bEarnFi,inte,"Deposit, EmergencyWithdraw Profit of Previous Iteration: 3.9000000000000004",50837.40108734835,-1.0,Unclassified,Pattern not matched																
8	bEarnFi,inte,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 12329",53931.984375,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
9	bEarnFi,inte,"Deposit, EmergencyWithdraw Profit of Previous Iteration: 4.6000000000000005",77117.9298022911,-1.0,Unclassified,Pattern not matched																
10	bEarnFi,inte,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 10061",71619.01345126051,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
11	bEarnFi,inte,"Deposit, EmergencyWithdraw Profit of Previous Iteration: 3.5",77975.33527393639,-1.0,Unclassified,Pattern not matched																
12	bEarnFi,poly,"Deposit, EmergencyWithdraw",-0.9999999850988388,-1.0,Unclassified,Pattern not matched																
13	bEarnFi,poly,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw",-0.9999999850988388,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
14	bEarnFi,poly,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 9702",-0.9999999850988388,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
15	bEarnFi,poly,"Deposit, EmergencyWithdraw Profit of Previous Iteration: 0.0",-0.9999999850988388,-1.0,Unclassified,Pattern not matched																
16	bEarnFi,poly,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 9702",-0.9999999850988388,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
17	bEarnFi,poly,"Deposit, EmergencyWithdraw Profit of Previous Iteration: 0.0",203508.9076014608,-1.0,Unclassified,Pattern not matched																
18	bEarnFi,poly,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 13257",207748.72773975134,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
19	bEarnFi,poly,"Deposit, EmergencyWithdraw Profit of Previous Iteration: 12.600000000000001",209868.63780888915,-1.0,Unclassified,Pattern not matched																
20	bEarnFi,poly,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 13770",203101.8780437708,-1.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
21	bEarnFi,poly,"Deposit, EmergencyWithdraw Profit of Previous Iteration: 15.200000000000001",207333.2053975016,4458.0,Unclassified,Pattern not matched																
22	bEarnFi,poly,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 13136",203508.9076014608,9702.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
23	bEarnFi,poly,"Deposit, EmergencyWithdraw, Deposit, EmergencyWithdraw Profit of Previous Iteration: 13715",207748.72773975134,2754.0,CWE-841,Reentrancy-like Pattern in Call Sequence																
24	CheeseBank,inte,"SwapUniswapETH2Cheese, SwapUniswapCheese2ETH",-443.0683038601477,-443.2,CWE-704,Incorrect Type Conversion or Cast																

**N.B.** Di fianco, è indicato un estratto del file .csv realizzato.

# Classificazione CWE – MITRE (1/2)

- Per la classificazione delle **CWE** indicate nel file .csv, si è fatto riferimento a quelle riportate sul sito del **MITRE** (<https://cwe.mitre.org/>).
- Per ottenere *informazioni aggiuntive* sulla vulnerabilità individuata, si scrive il corrispondente **ID** all'interno della box «**ID Lookup**».





# Classificazione CWE – MITRE (2/2)

1. **CWE-841** – *Reentrancy-like Pattern in Call Sequence* → rappresenta attacchi in cui una funzione viene richiamata ricorsivamente in modo inaspettato, tipico nei *bug di reentrancy*.
2. **CWE-704** – *Incorrect Type Conversion or Cast* → converte dati da un tipo all'altro in modo errato, con possibili effetti imprevedibili (overflow, perdita di precisione).
3. **CWE-639** – *Authorization Bypass Through User-Controlled Key* → accesso non autorizzato a risorse o dati modificando identificatori controllati dall'utente (es. ID di un account).
4. **CWE-346** – *Origin Validation Error* → mancata verifica dell'origine di una richiesta o chiamata (es. una funzione skim o sync invocata fuori contesto).
5. **Undefined** → lo script non ha trovato un pattern chiaro nella sequenza di funzioni del contratto per assicurare una CWE specifica.

# CWE-841

Facendo riferimento alla documentazione ufficiale, notiamo che **CWE-841** – *Reentrancy-like Pattern in Call Sequence* corrisponde a *Improper Enforcement of Behavioral Workflow*.

## CWE-841: Improper Enforcement of Behavioral Workflow

Weakness ID: 841

Vulnerability Mapping: ALLOWED

Abstraction: Base

- È una vulnerabilità in cui un'applicazione non impone correttamente l'ordine o la sequenza delle azioni richieste da un attore (utente, contratto, processo, ecc.).
- Questo consente ad attori malevoli di eseguire azioni (es. Deposit, Withdraw) in una **sequenza anomala**, portando a comportamenti logici imprevisti. È alla base di molti attacchi di tipo **reentrancy** e **logic manipulation**.
- Notiamo, dunque, che il nome assegnato tramite lo script python rimane coerente con quanto indicato nella documentazione.

# CWE-704

Facendo riferimento alla documentazione ufficiale, notiamo che **CWE-704** – *Incorrect Type Conversion or Cast* corrisponde proprio a quanto indicato cercando l'ID corrispondente.

## CWE-704: Incorrect Type Conversion or Cast

Weakness ID: 704

Vulnerability Mapping: **ALLOWED** (with careful review of mapping notes)

Abstraction: Class

- È una vulnerabilità che si verifica quando un'applicazione converte in modo errato un oggetto, risorsa o struttura da un tipo a un altro, portando a **comportamenti imprevisti o pericolosi**.
- Nel contesto di FlashSyn, la CWE-704 può manifestarsi quando, ad esempio, una funzione di un contratto intelligente interpreta erroneamente un valore numerico a causa di una **conversione di tipo non sicura**, portando a calcoli di profitto errati o a vulnerabilità sfruttabili.

# CWE-639

Facendo riferimento alla documentazione ufficiale, notiamo che **CWE-639** – *Authorization Bypass Through User-Controlled Key* corrisponde proprio a quanto indicato cercando l'ID corrispondente.

## CWE-639: Authorization Bypass Through User-Controlled Key

Weakness ID: 639

Vulnerability Mapping: ALLOWED

Abstraction: Base

- È una vulnerabilità che si verifica quando un sistema consente a un utente di accedere o modificare dati di altri utenti manipolando direttamente un identificatore (come un ID numerico) senza un'adeguata verifica dei permessi.
- Solitamente, può portare a **escalation dei privilegi, violazione della privacy, accesso ad informazioni sensibili** con relativa modifica o cancellazione.

# CWE-346

Facendo riferimento alla documentazione ufficiale, notiamo che **CWE-346** – *Origin Validation Error* corrisponde proprio a quanto indicato cercando l'ID corrispondente.

## CWE-346: Origin Validation Error

Weakness ID: 346

Vulnerability Mapping: **ALLOWED** (with careful review of mapping notes)

Abstraction: Class

- È una vulnerabilità che si verifica quando un'applicazione non verifica correttamente l'origine di una richiesta o di un dato, consentendo a fonti non autorizzate di interagire con il sistema.
- Nel contesto della DeFi e delle blockchain, questa vulnerabilità è particolarmente rilevante per gli **smart contract**, i quali si aspetterebbero chiamate solo da contratti fidati, senza imporre alcun controllo.

# CVE - Common Vulnerabilities and Exposures

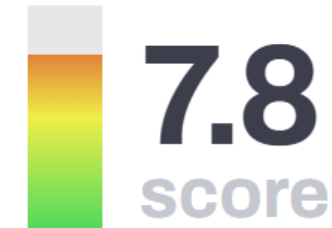
CWE	CVE	Descrizione	Rilevanza per FlashSyn
<b>CWE-841</b> Improper Enforcement of Behavioral Workflow	<i>Nessuna CVE specifica trovata</i>	Questa CWE si verifica quando un sistema non impone correttamente l'ordine delle operazioni, permettendo azioni in sequenza non autorizzate.	FlashSyn può identificare sequenze di funzioni nei contratti DeFi che violano l'ordine previsto, simulando potenziali exploit.
<b>CWE-704</b> Incorrect Type Conversion or Cast	<a href="#">CVE-2021-35091</a>	Vulnerabilità in Qualcomm Snapdragon dovuta a conversioni di tipo errate durante la gestione della memoria, portando a letture fuori dai limiti.	Sebbene non direttamente correlato a DeFi, evidenzia i rischi associati a conversioni di tipo errate, che FlashSyn può simulare nei contratti intelligenti.
<b>CWE-639</b> Authorization Bypass Through User-Controlled Key	<a href="#">CVE-2023-48783</a>	Vulnerabilità in FortiPortal che consente a utenti autenticati di accedere a dati di altre organizzazioni modificando parametri controllati dall'utente.	FlashSyn può rilevare casi in cui contratti DeFi permettono l'accesso non autorizzato a dati o funzioni tramite parametri manipolabili.
<b>CWE-346</b> Origin Validation Error	<a href="#">CVE-2023-46715</a>	Errore di validazione dell'origine in FortiOS IPsec VPN che permette a utenti autenticati di inviare pacchetti con IP spoofato.	FlashSyn può simulare scenari in cui contratti DeFi non verificano correttamente l'origine delle richieste, esponendosi a potenziali attacchi.

# CVE-2021-35091 (1/2)

- È una vulnerabilità presente nei prodotti **Snapdragon Connectivity** e **Snapdragon Mobile** di Qualcomm. Il **problema** deriva da una conversione di tipo impropria (typecasting errato) durante la gestione dei **page fault** relativi alla memoria globale, che può portare a una lettura di memoria fuori dai limiti (**out of bounds**).
- Questo potrebbe permettere a un attaccante di **accedere a informazioni sensibili** conservate nella memoria, che dovrebbero invece essere inaccessibili.
- Gli amministratori di sistema potrebbero applicare delle **patch**, come:
  - Aggiornare il firmware dei dispositivi
  - Applicare eventuali aggiornamenti di sistema
  - Monitorare gli avvisi di sicurezza Qualcomm per ulteriori informazioni.

**N.B.** <https://cve.armis.com/cve-2021-35091> (Armis).

## CVE-2021-35091



Published Date: Jun 14, 2022

Industry Exposure



Severity  
**High**

CWE ID: CWE-704

This does not appear in CISA KEV. Would you like to know if Armis has Early Warning alerts? [Learn More](#)


# CVE-2021-35091 (2/2)

## Threat Predictions

EPSS Score  
**0.1**

EPSS Percentile  
**32%**

## Exploitability

Score  **1.8**

Attack Vector Local


Attack Complexity Low

Privileges Required Low

User Interaction None

Scope Unchanged

## Impact

Score  **5.9**

Confidentiality Impact High

Integrity Impact High

Availability Impact High

**N.B.** <https://cve.armis.com/cve-2021-35091> (Armis).



# CVE-2023-48783

- Questa vulnerabilità può consentire a un *utente remoto e autenticato*, anche con soli **permessi in sola lettura**, di accedere agli *endpoint di altre organizzazioni* inviando **richieste GET appositamente costruite**.
- Una vulnerabilità di tipo "*Authorization Bypass Through User-Controlled Key*" (**CWE-639**) riguarda **PortiPortal** in differenti versioni (**7.2.1 e precedenti**).
- Si possono adottare **patch** come: controlli lato server, restrizione e validazione degli endpoint, e stabilire un'associazione stretta tra sessione e risorsa.

**Metrics**

CVSS Version 4.0 CVSS Version 3.x CVSS Version 2.0

*NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.*

**CVSS 3.x Severity and Vector Strings:**

 **CNA:** Fortinet, Inc.

**Base Score:** 5.4 MEDIUM

**Vector:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N

**N.B.** <https://nvd.nist.gov/vuln/detail/CVE-2023-48783> (NIST).



# CVE-2023-46715

- Questa vulnerabilità consente a un *utente autenticato della VPN IPsec* con indirizzamento IP dinamico di **inviare (ma non ricevere)** pacchetti che **imitano l'indirizzo IP di un altro utente**, sfruttando pacchetti di rete appositamente creati.
- Una vulnerabilità di tipo *Origin Validation Error* (**CWE-346**) è stata identificata nei prodotti **Fortinet FortiOS IPsec VPN**, nelle versioni dalla **7.4.0 alla 7.4.1** e nella versione **7.2.6 e precedenti**.
- **Fortinet** ha rilasciato aggiornamenti per correggere questa vulnerabilità.

**Metrics** CVSS Version 4.0 CVSS Version 3.x CVSS Version 2.0

*NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.*

**CVSS 3.x Severity and Vector Strings:**

	<b>NIST:</b> NVD	<b>Base Score:</b> 4.3 MEDIUM	<b>Vector:</b> CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N
	<b>CNA:</b> Fortinet, Inc.	<b>Base Score:</b> 5.0 MEDIUM	<b>Vector:</b> CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:L/A:N

**N.B.** <https://nvd.nist.gov/vuln/detail/CVE-2023-46715> (NIST).

# Conclusioni sull'analisi CWE/CVE

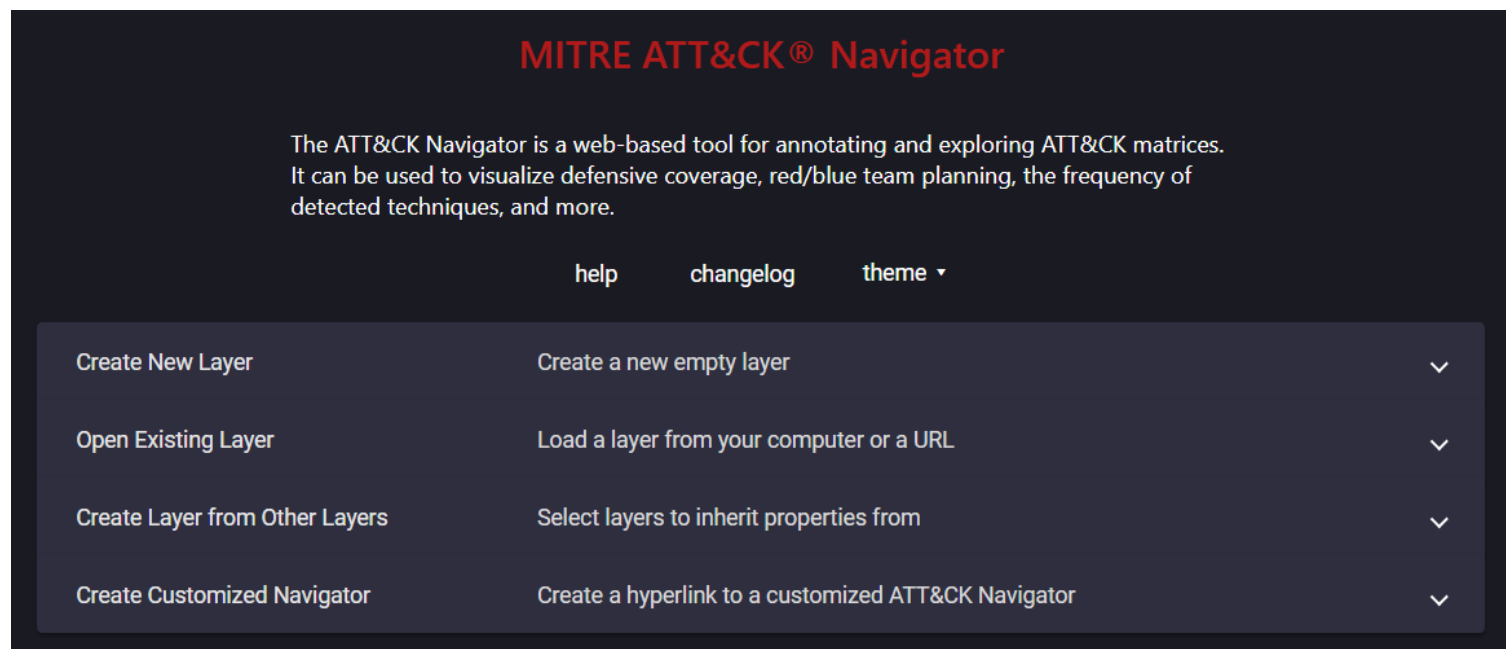
L'analisi CWE/CVE del progetto FlashSyn ha permesso di:

- *Classificare* in modo formale le **vulnerabilità** identificate da FlashSyn.
- *Collegare* le **debolezze** simulate a casi reali documentati (es. CVE-2021-35091, CVE-2023-48783).
- Evidenziare **vulnerabilità comuni nei protocolli DeFi**, come accessi non autorizzati, errori di conversione e mancanza di validazioni.

Questo approccio favorisce una maggiore interoperabilità tra sicurezza tradizionale e smart contracts, e valorizza l'uso di standard consolidati come CWE e CVE per la classificazione e mitigazione delle vulnerabilità.

# Mapping MITRE ATT&CK – Navigator (1/2)

- Le vulnerabilità identificate con FlashSyn sono state **mappate su tecniche MITRE ATT&CK** per evidenziare i possibili *percorsi di attacco* che un attore malintenzionato potrebbe intraprendere. Questo approccio consente di passare da una *visione “code-level”* a una *visione “threat-level”*.
- Per effettuare il mapping è stato utilizzato il Navigator del MITRE (<https://mitre-attack.github.io/attack-navigator/>).



# Mapping MITRE ATT&CK – Navigator (2/2)

CWE	Descrizione	Tecnica ATT&CK	Codice	Tattica
CWE-841	Sequenze impreviste di chiamate	Abuse Elevation Control Mechanism	T1548	Privilege Escalation
CWE-704	Conversione tipo errata → crash o errore logico	Endpoint Denial of Service: Application or System Exploitation	T1499.004	Impact
CWE-639	Accesso non autorizzato via chiavi utente	Valid Accounts	T1078	Initial Access
		Permission Groups Discovery	T1069	Discovery
CWE-346	Mancata verifica dell'origine	Exploitation for Privilege Escalation	T1068	Privilege Escalation
		Impersonation	T1656	Defense Evasion