Hunting and Monitoring for On-Chain Attacks

Dmitry Gusakov
Automation Lead, Lido

Christian Seifert
Researcher-in-Residence, Forta Foundation
About us

Automation Lead at **Lido**
Building solid on-chain and off-chain monitoring tools
Responsible for the quality of major Lido products

Prior to Lido worked for 6 years as a **Quality and Automation Engineer**
Quality assurance
Security testing

[@d_gusakov](https://twitter.com/d_gusakov)  [@Gusakov_dv](https://twitter.com/Gusakov_dv)

---

PhD in Computer Science from Victoria University of Wellington, NZ

Researcher-in-Residence at **the Forta Foundation**
Analyzing attacks
Working with community to strengthen attack detection

Prior to Forta worked at **Microsoft** for 14 years on the Defender product line

Security Research
Data Science

[@cseifert](https://twitter.com/cseifert)  [@christian_forta](https://twitter.com/christian_forta)
Web3 is getting hacked!
Tornado Cash Funding

Suspicious Contract Creation

Flashloan Flashbot Tx

Tornado Cash Money Laundering
Tornado Cash Funding
Jun-16-2022 08:45:36 AM +UTC - https://etherscan.io/address/0x7b792e49f640676b3706d666075e903b3a4dec6#internaltx

Address: 0x7b792E49f640676b3706d666075E903B3A4dec6

<table>
<thead>
<tr>
<th>Parent Txn Hash</th>
<th>Block</th>
<th>Age</th>
<th>From</th>
<th>To</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x84ee1ce4ddd2aa5113a...</td>
<td>14972410</td>
<td>106 days 12 hrs ago</td>
<td>Tornado.Cash: 1 ETH</td>
<td>Inverse Finance Exploiter</td>
<td>0.975623 Ether</td>
</tr>
</tbody>
</table>
## Suspicious Contract Creation

Jun-16-2022 08:47:50 AM +UTC - [https://etherscan.io/tx/0xfb5a4d1ae9f8458f673f301c2e713613662ad621e8f57065a4da58a6401c0b4d](https://etherscan.io/tx/0xfb5a4d1ae9f8458f673f301c2e713613662ad621e8f57065a4da58a6401c0b4d)

### Transactions

<table>
<thead>
<tr>
<th>Txn Hash</th>
<th>Method</th>
<th>Block</th>
<th>Age</th>
<th>From</th>
<th>To</th>
<th>Value</th>
<th>Txn Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xfb5a4d1ae9f8458f673f301c2e713613662ad621e8f57065a4da58a6401c0b4d</td>
<td>OUT</td>
<td>14972418</td>
<td>106 days 12 hrs ago</td>
<td>Inverse Finance Explorer</td>
<td>Contract Creation</td>
<td>0 Ether</td>
<td>0.11407894</td>
</tr>
</tbody>
</table>

### Sponsored:
- 1 - Inch - The most efficient DEX aggregator. Recover up to 95% of gas spendings. **Swap now!**

A total of 38 transactions found.
# Flashloan/Flashbot Usage

Jun-16-2022 08:47:58 AM +UTC - https://etherscan.io/address/0x7b792e49f640676b3706d66075e903b3a4deec6#internaltx

## Transaction Details

<table>
<thead>
<tr>
<th>Feature</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Hash</td>
<td>0x958236266991bc3fe3b77feaec1a120f172c0708ad01c7a715b255f218f9313c</td>
</tr>
<tr>
<td>Status</td>
<td>✔ Success</td>
</tr>
<tr>
<td>Block</td>
<td>✔ 14972419 (678319 Block Confirmations)</td>
</tr>
<tr>
<td>Timestamp</td>
<td>106 days 12 hrs ago (Jun-16-2022 08:47:58 AM +UTC)</td>
</tr>
<tr>
<td>Transaction Action</td>
<td>Flash Loan 27,000 ➤ WBTC From Aave Protocol V2</td>
</tr>
</tbody>
</table>

*Featured: Bridging tokens between Ethereum, Layer 2 and other chains? Browse through the Blockscan bridges list.*
### Exploit Impact

Jun-16-2022 08:47:50 AM +UTC - https://phalcon.blocksec.com/tx/eth/0x958236266991bc3fe3b77f3aacea120f172c0708ad01c7a715b255f218f9313c

<table>
<thead>
<tr>
<th>Address</th>
<th>Token</th>
<th>Balance</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xf5088c58ce57ce40a40997c715075172691f92e2d</td>
<td>anYvCrv3Crypto, USDT, WBTC</td>
<td>+245,337.73387519, +99,976.294967, +53,244,6541</td>
<td>-</td>
</tr>
<tr>
<td>anYvCrv3Crypto</td>
<td>yvCurve-3Crypto</td>
<td>+4,986,7546775039741431</td>
<td>-</td>
</tr>
<tr>
<td>anYvCrv3Crypto</td>
<td>anYvCrv3Crypto</td>
<td>-245,337.73387519</td>
<td>-</td>
</tr>
<tr>
<td>Curve.fi: DAI/USDC/USDT Pool</td>
<td>USDT</td>
<td>-10,899,976.294967</td>
<td>-</td>
</tr>
<tr>
<td>0x464c71f6c2f760dda6093dcb91c24c39e5d6e18c</td>
<td>aWBTC</td>
<td>+0.00004148</td>
<td>-</td>
</tr>
<tr>
<td>aWBTC</td>
<td>WBTC</td>
<td>+24.5</td>
<td>-</td>
</tr>
<tr>
<td>0xda51a-Vyper_contract</td>
<td>WBTC, USDT</td>
<td>-77,544,6541, +10,000,000</td>
<td>-</td>
</tr>
</tbody>
</table>
## Money Laundering

Jun-16-2022 08:47:58 AM +UTC - [https://etherscan.io/tx/0x37e015682d3d989a90f7e47ee0c12a3bc58a96a671b6eeb9691e03e79ac179d4](https://etherscan.io/tx/0x37e015682d3d989a90f7e47ee0c12a3bc58a96a671b6eeb9691e03e79ac179d4)

### Transaction Details

**Overview**

<table>
<thead>
<tr>
<th>Transaction Hash</th>
<th>Status</th>
<th>Block</th>
<th>Timestamp</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x37e015682d3d989a90f7e47ee0c12a3bc58a96a671b6eeb9691e03e79ac179d4</td>
<td>Success</td>
<td>14972437</td>
<td>106 days 12 hrs ago</td>
<td>Confirmed within 1 min</td>
</tr>
</tbody>
</table>

**Transaction Action**

- Swap 42.59557232 WBTC For 785.666048274022634622 Ether On Uniswap V3
- Swap 7.98666981 WBTC For 148.22480221160635588 Ether On Uniswap V3
- Swap 2.66222327 WBTC For 57,811.229075 USDC On Uniswap V3
- Swap 57,811.229075 USDC For 49.413422222684903223 Ether On Uniswap V3
### Money Laundering

Jun-16-2022 08:56:47 AM +UTC - https://etherscan.io/tx/0xf9953c26d229c42938f681ce348322c92a5178965a6631a0f09fcadbac16a9d7
Jun-16-2022 08:56:47 AM +UTC - https://etherscan.io/tx/0xec27c61ae0c5a3f3f8a48bbb7b1f38781205ee1b8a978ee83e0b512c1bb6e22b

#### Transactions

<table>
<thead>
<tr>
<th>Txn Hash</th>
<th>Method</th>
<th>Block</th>
<th>Age</th>
<th>From</th>
<th>To</th>
<th>Value</th>
<th>Txn Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0426bec92942e576ab...</td>
<td>Deposit</td>
<td>1497247</td>
<td>106 days 11 hrs ago</td>
<td>Inverse Finance Exploiter</td>
<td>Inverse Finance Exploiter</td>
<td>100 Ether</td>
<td>0.02585584</td>
</tr>
<tr>
<td>0x70770bb62c9838b0a04a5...</td>
<td>Deposit</td>
<td>1497234</td>
<td>106 days 11 hrs ago</td>
<td>Inverse Finance Exploiter</td>
<td>Inverse Finance Exploiter</td>
<td>100 Ether</td>
<td>0.02673976</td>
</tr>
<tr>
<td>0xe694481bab325997b5b23...</td>
<td>Transfer Tokens</td>
<td>1497236</td>
<td>106 days 11 hrs ago</td>
<td>Inverse Finance Exploiter</td>
<td>Inverse Finance Exploiter</td>
<td>0 Ether</td>
<td>0.00365010</td>
</tr>
<tr>
<td>0x0e024ae1ce47f33458a7c...</td>
<td>Deposit</td>
<td>1497236</td>
<td>106 days 11 hrs ago</td>
<td>Inverse Finance Exploiter</td>
<td>Inverse Finance Exploiter</td>
<td>100 Ether</td>
<td>0.0258806</td>
</tr>
<tr>
<td>0xe2c7c61ae0c5a3f3f8a48bb...</td>
<td>Approve</td>
<td>1497268</td>
<td>106 days 11 hrs ago</td>
<td>Inverse Finance Exploiter</td>
<td>Inverse Finance Exploiter</td>
<td>0 Ether</td>
<td>0.00240049</td>
</tr>
</tbody>
</table>
Comprehensive security strategy

Pre-deployment
- Template contracts
- Audit

Post-deployment
- Bug bounties
- Real-time monitoring and alerting
- Incident/emergency response
Decentralized security camera and alarm system for Web 3
Detection Bots
“Security cameras”

A script (piece of logic) that any developer can write and publish to the Forta Network.

Detection bots tell the network what to watch.

Scan Node
“Alarm system”

Runs the detection bots against each block of transactions.

The nodes power the detection bots, and keep them running 24/7.
Forta Explorer (link)

The Forta Network Explorer

Network Activity

Information about the health and activity of the Forta Network, including blockchain coverage and active node operators.
**Attack Stages**

**Funding**
- TC funding
- Exchange funding
- New account
- Bridge funding

**Preparation**
- Sleep minting
- Attack contract creation
- Ice Phishing
- Token Approvals
- Token impersonation

**Exploitation**
- Flashloan price manipulation
- Flashbot usage
- Ice Phishing
- Token Transfers
- Rug pulls
- Exploit (reentrancy, failed access control, etc.)

**Money Laundering**
- TC deposits
- Exchange deposits
- Exchange into native tokens
- Bridge deposits
- Wash trading
Flash Loan

BotID: 0x55636f5577694c83b84b0687eb77863850c50bd9f6072686c8463a0cbc5566e0
Attack Stage: Exploitation

What is It?

Flash loans are temporary loans (has to be paid back in one transaction) that allows borrowers to obtain large amount of tokens. This is used, for example, for arbitrage.

Flash loans are also utilized by attackers to manipulate prices temporarily. This could be used to exploit a vulnerable protocol (e.g. by taking out an undercollateralized loan).

How to detect?

Identify all transactions obtaining a flash loan
Assess whether profit exceeds a particular threshold.
Rug Pulls
BotID: 0x580d14bed37f523d14edcfa83ae87e168ac333a98f70c4f9991357e1b7ee855f
Attack Stage: Exploitation

What is It?
Rug pulls are tokens that are hyped by creators. As they are traded on DEXes, creators may dump existing tokens or dump newly created tokens. The price crashes and remaining token holders are left holding the bag of worthless tokens.

How to detect?
Obtain price information from DEXes (utilizing common ABI)
Trigger on significant price drops.
Price fluctuations are common, however. In order to reduce noise, one needs to apply a time series anomaly detection approach that takes into account historical information (seasonality, volatility).
Exploit Simulation

BotID: 0xe8527df509859e531e58ba4154e9157eb6d9b2da202516a66ab120deabd3f9f6
Attack Stage: Preparation

What is It?
For certain attacks (e.g. economic attacks, reentrancy attacks), attackers need to deploy a smart contract. Several indicators can help to determine whether a smart contract is malicious (e.g. was it created through Tornado Cash funded EOA; is it verified on Etherscan?)

These contracts contain all the code that is needed to execute the exploit.

How to detect?
Upon smart contract deployment, locally fork the chain using Ganache. Invoke all exposed functions (essentially fuzzing the smart contract).

Assess whether large amounts of tokens are transferred into the attacker’s wallet or contract.

Identifies the attack before it is executed on-chain.
**Custom Bot Development**

**Define the Requirements**

**Step 1**

Based on threat model define the requirements:
- What is the logic?
- What alerts will the bot emit?
- What data do you need?
- What chains should the bot run on? What differences exist between the chains?

**Implement and Test**

**Step 2**

Implement using Python/JavaScript/TypeScript SDK

Test using unit tests

Test retroactively on existing attack transactions/ blocks

Execute locally against live transaction feed

**Deploy and Subscribe**

**Step 3**

Deploy in a permissionless way to the Forta Network using the CLI or Forta App. It will be deployed onto several nodes to create redundancy and increase alert reliability.

Log and alert data for the bot can be viewed and monitored through bot stats page.

Alert subscriptions can be configured to receive alerts on Telegram, Slack, Discord and accessed through the GraphQL API.

Get started at:

**Setup Dev Environment**

**Step 1**

- Node.js v12+ (which includes the Node package manager i.e. npm)
- Conda & Python v3.6+ (only if you want to use Python SDK)
- Docker v20+
Custom Bot Development

Initialize Bot
Step 2

$ mkdir my-new-bot
$ cd my-new-bot
$ npx forta-agent@latest init
  --python

Main Bot Logic
Custom Bot Development

**Initialize Bot**  
Step 2

```
$ mkdir my-new-bot  
$ cd my-new-bot  
$ npx forta-agent@latest init --python
```

Unit tests
Custom Bot Development

**Initialize Bot**

**Step 2**

```
$ mkdir my-new-bot
$ cd my-new-bot
$ npx forta-agent@latest init
--python
```

Docker build file

```
. dockerignore
.gitignore

Dockerfile

- forta.config.json
- package-lock.json
- package.json
- pytest.ini
- README.md
- requirements_dev.txt
- requirements.txt
```
Custom Bot Development

Initialize Bot
Step 2

$ mkdir my-new-bot
$ cd my-new-bot
$ npx forta-agent@latest init --python
Custom Bot Development

Create Documentation
Step 3

Capture
- Title
- Description
- Supported Chains
- Alerts
- Test Data

```python
# Large Tether Transfer Agent

## Description
This agent detects transactions with large Tether transfers

## Supported Chains
- Ethereum
- List any other chains this agent can support e.g. BSC

## Alerts
Describe each of the type of alerts fired by this agent
- FORTA-1
  - Fired when a transaction contains a Tether transfer over 10,000 USDT
  - Severity is always set to "low" (mention any conditions where it could be something else)
  - Type is always set to "info" (mention any conditions where it could be something else)
  - Mention any other type of metadata fields included with this alert

## Test Data
The agent behaviour can be verified with the following transactions:
- 0x3a0f757030beec55c22cbc545dd8a844cbbb2e6019461769e1bc3f3a95d10826 (15,000 USDT)
```
Custom Bot Development

Implement Bot Logic
Step 4

- Process Tx
- Filter for USDT events
- Normalize value
- Assess against threshold
- Emit alert

def handle_transaction(transaction_event) -> list:
    findings = []

    # limiting this agent to emit only 5 findings so that the alert feed is not spamme
global findings_count
if findings_count >= 5:
    return findings
Custom Bot Development

Implement Bot Logic
Step 4

- Process Tx
- Filter for USDT events
- Normalize value
- Assess against threshold
- Emit alert

```python
def handle_transaction(transaction_event) -> list:
    findings = []

    # limiting this agent to emit only 5 findings so that the alert feed is not spammy
    global findings_count
    if findings_count >= 5:
        return findings

    findings.append(Finding(
        'name': 'High Tether Transfer',
        'description': f'High amount of USDT transferred: \{normalized_value\}',
        'alert_id': 'FORTA-1',
        'severity': FindingSeverity.Low,
        'type': FindingType.Info,
        'metadata': {
            'to': to,
            'from': from_,
        }
    ))
    findings_count += 1

    return findings
```

Returns a list of findings
Custom Bot Development

Implement Bot Logic
Step 4

- Process Tx
- Filter for USDT events
- Normalize value
- Assess against threshold
- Emit alert

```python
def handle_transaction(transaction_event) -> list:
    findings = []

    # limiting this agent to emit only 5 findings so that the alert feed is not spamme
    global findings_count
    if findings_count >= 5:
        return findings

    findings.append(Finding({
        'name': 'High Tether Transfer',
        'description': f'High amount of USDT transferred: {normalized_value}',
        'alert_id': 'FORTA-1',
        'severity': FindingSeverity.Low,
        'type': FindingType.Info,
        'metadata': {
            'to': to,
            'from': from_,
        }
    }))
    findings_count += 1

    return findings
```

Findings contain all pertinent information
Custom Bot Development

Implement Bot Logic
Step 4

- Process Tx
- Filter for USDT events
- Normalize value
- Assess against threshold
- Emit alert

**Event Filter**

```python
# filter the transaction logs for any Tether transfers
ERC20_TRANSFER_EVENT = ' côt":{"name":"Transfer","type":"event","anonymous":false,"inputs":
TETHER_ADDRESS = '0xdaC17F958D2ee523a2206206994597C13D831ec7'
TETHER_DECIMALS = 6

tether_transfer_events = transaction_event.filter_log(ERC20_TRANSFER_EVENT, TETHER_ADDRESS)

for transfer_event in tether_transfer_events:
    # extract transfer event arguments
    to = transfer_event['args']['to']
    from_ = transfer_event['args']['from']
    value = transfer_event['args']['value']
    # shift decimals of transfer value
    normalized_value = value / 10 ** TETHER_DECIMALS

    # if more than 10,000 Tether were transferred, report it
    if normalized_value > 10000:
        findings.append(Finding({
```
Custom Bot Development

Implement Bot Logic
Step 4

- Process Tx
- Filter for USDT events
- Normalize value
- Assess against threshold
- Emit alert

```python
# filter the transaction logs for any Tether transfers
tether_transfer_events = transaction_event.filter_log(
    ERC20_TRANSFER_EVENT, TETHER_ADDRESS)

for transfer_event in tether_transfer_events:
    # extract transfer event arguments
    to = transfer_event['args']['to']
    from_ = transfer_event['args']['from']
    value = transfer_event['args']['value']

    # shift decimals of transfer value
    normalized_value = value / 10 ** TETHER_DECIMALS

    # if more than 10,000 Tether were transferred, report it
    if normalized_value > 10000:
        findings.append(Finding({})
```
Custom Bot Development

Implement Bot Logic
Step 4

- Process Tx
- Filter for USDT events
- Normalize value
- Assess against threshold
- Emit alert

```python
# filter the transaction logs for any Tether transfers
TETHER_ADDRESS = '0x0dAB5F95B5b2f0D8C878828e523a2206206994597C13D831e7c'
TETHER_DECIMALS = 6

ERC20_TRANSFER_EVENT = '{"name":"Transfer","type":"event","anonymous":false,"inputs":[]}

# if more than 10,000 Tether were transferred, report it
if normalized_value > 10000:
    findings.append(Finding({
        'description': 'Tether transfer above threshold',
        'selector': 'ERC20_TRANSFER_EVENT',
        'address': TETHER_ADDRESS,
        'value': normalized_value,
        'block_number': block_number
    }))
```

Emit finding
Custom Bot Development

Test, test, test
Step 5

- Unit test
- Backtest
- Live test

> forta-agent-starter@0.0.1 test
> python3 -m pytest

============================================= test session starts =============================================
platform darwin -- Python 3.8.13, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: /Users/christianseifert/forta/tutorial/my-new-bot, configfile: pytest.ini
plugins: env-0.6.2, web3-5.23.0
collected 2 items

src/agent_test.py ..

============================================= 2 passed in 0.11s ==============================================

> (forta) christianseifert@x86_64-apple-darwin13 my-new-bot %
Custom Bot Development

Test, test, test

Step 5

- Unit test
- Backtest
- Live test

# Test Data

The agent behaviour can be verified with the following transactions:

- 0x3a0f757030beec55c22cbc545dd8a844cbbb2e6019461769e1bc3f3a95d10826 (15,000 USDT)
- (forta) christianseifert@x86_64-apple-darwin13 my-new-bot % npm run tx 0x3a0f757030beec55c22cbc545dd8a844cbbb2e6019461769e1bc3f3a95d10826

> forta-agent-starter@0.0.1 tx
> forta-agent run --tx 0x3a0f757030beec55c22cbc545dd8a844cbbb2e6019461769e1bc3f3a95d10826

1 findings for transaction 0x3a0f757030beec55c22cbc545dd8a844cbbb2e6019461769e1bc3f3a95d10826 {
  "name": "High Tether Transfer",
  "description": "High amount of USDT transferred: 15000.0",
  "alertId": "FORTA-1",
  "protocol": "ethereum",
  "severity": "Low",
  "type": "Info",
  "metadata": {
    "to": "0x191a95daC026F3A002C66e6C61C484FAb9D65D17",
    "from": "0x02f475Ce4490CAffFEA57f3ab0F7D7831D6B1fC6"
  },
  "addresses": []
}
Custom Bot Development

Test, test, test
Step 5
- Unit test
- Backtest
- Live test

fetching block 5677316...

0 findings for transaction 0xeeee697d5b82e2351dc301a8e66fba58b85298ef8e38e6ff7144c33961137a8f2530
0 findings for transaction 0xae33ad4a9e4b503e54f5eb4de2e0557962afbf8c891348b8c191c921d4820
0 findings for transaction 0x7a98b03411eef87627c555e46afe29e3e2400000e90587b29a2608d7523c2
0 findings for transaction 0x4f9cd92f3c39927cb5267faba1208262a3284a8958caf6b15ee9ac03989a9bedad
0 findings for transaction 0xda2f21b6a36d6c37aeb7b79cbb7e7fc664e53b80bdc3693093500c172e20b7
0 findings for transaction 0xe9b967823e1a846b13a76907faeaab773adb2d7263575468336c193c03a13b0182d1a
0 findings for transaction 0x4ae4b2375b527d18dbd535cc5d2e96bcde2383d07504ca3da70deb7ca2e0de
0 findings for transaction 0x7bd1baf996aa60c1718e8eb78b949ce0abb8cfd08e7bc0f7f1c8f89cf5c7f8ce38f
0 findings for transaction 0x791163dc5e09ebe289389fd11088db1779ce7927e4be42018f33f28c1aff58e7
0 findings for transaction 0xac6643b920a3af882139566df17f33b06fe789d7952358343fb03e6f5c3ad
0 findings for transaction 0xda7c1c89297069d6a98d58cc9ca6df77ba1b0ed38d78a2e9d2a95b88b5d61738
0 findings for transaction 0xe6bf7b38e11173d35b186884a22c8f09b6b8909f07c0e872ace1659535c4f4
0 findings for transaction 0xa4f5e542b9a6a652155a5e67d271b41f580d3e6f8f2c857fa51aee5e419922
0 findings for transaction 0x87f8b7f38e11173d35b186884a22c8f09b6b8909f07c0e872ace1659535c4f4
0 findings for transaction 0x9f0738b7f46067a7940b6ff43932d8bdc6b5e5ea54e870498e84d29a64572c
0 findings for transaction 0x87f085873ee461c7989e9ed21f66aeec9c3e66144018a3c3e67957edadb5094c
0 findings for transaction 0xc8f48c5b5991f1d4a34543f5ab7dc9bcb5a4b31f59c5f9d77ad4068b7349a80b80
0 findings for transaction 0xf52fab23f84f06be4c9e28f06669c0a6720a13b34a324489a701fbb0f4214e3c
0 findings for transaction 0x1408be7900923842db6e8d7471f57c78cbde9e70e36b50f256d712447e42e1
Custom Bot Development

Deploy
Step 6
- Deploy using CLI
  - npm run publish
- Need some MATIC
Custom Bot Development

Deploy

Step 6

- Deploy using CLI
  - npm run publish
- Need some MATIC
Custom Bot Development

Subscribe
Step 7

- Various mechanisms:
  - Telegram
  - Email
  - Slack
  - Discord
  - Webhook
Custom Bot Development

Subscribe
Step 7

- Various mechanisms:
  - Telegram
  - Email
  - Slack
  - Discord
  - Webhook
Bot Development Exercises

**Exercise 1**
Detect large USDC transfers
- Simple operational bot alerting on large (10K+) USDC transfers
- Filter for Events (ERC20_TRANSFER/USDC Token Contract)
- Normalize value
- Threshold on the value

**Exercise 2**
Detect low balances
- Bot to identify low balances. Allows to monitor your own address
- Assess balance with each block/tx
- Alert when it falls below a threshold
- Cache so you don’t receive alert barrage

**Exercise 3**
Flashloan resulting in losses in Yearn Dai Vault
- Assess whether flashloan protocol and Vault were touched in tx
- Assess whether flashloan was obtained
- Assess vault balance before and this block to derive difference
- Threshold on diff and alert

Get started at:
https://github.com/forta-network/forta-bot-workshop
Bot Development Contest

Context 9
Identify attacked protocol

- Win up to 3,000 USD
- Alerts today identify attacks and expose slew of addresses involved in the transaction. Post analysis needs to be performed to identify what protocol was attacked
- Bot’s goal is to identify the protocol attacked (e.g. by analyzing token transfers)

https://docs.forta.network/en/latest/contest9-forta/
Part 2.
Let’s look at the protocol code
# Why we need protocol alerting?

<table>
<thead>
<tr>
<th></th>
<th>Generic alerting</th>
<th>Dedicated alerting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow detection of generic attacks?</td>
<td>Yep</td>
<td>Kinda</td>
</tr>
<tr>
<td>Gives confidence in YOUR protocol safety</td>
<td>50 / 50</td>
<td>&gt; 90%</td>
</tr>
<tr>
<td>Main attack vectors detected</td>
<td>Both generic and specific attacks</td>
<td>Generic changes and uncertainties in the protocol, but not attacks itself</td>
</tr>
<tr>
<td>Set-up</td>
<td>You can use existing bots</td>
<td>You need to build a bot yourself</td>
</tr>
</tbody>
</table>
We need them ALL
Typical protocol alerts

Operations

Repetitive events

- Oracle reports
- Rewards distribution
- Funds deposits
- DAO Voting
- ...

Predictions of possible issues in operations

- Sloppy oracles
- Low balance of executors
- Unexpected funds movements
- Unexpected vote content
- ...

Forta LIDO
Typical protocol alerts

Security

Inconsistency in protocol invariants

- Bridge balance difference (bridge hack)
- Issuing of the tokens with no actual backup (protocol hack)
- Minting NFTs with no actual backup (protocol hack)
- Unexpected fund transfers (protocol ownership loss)
- ...

Typical protocol alerts

Security

Events that should not happen silently
- Huge withdrawals
- Huge balance changes
- ...

ACL changes
- Role granted/revoked
- Ownership transferred
- ...

Events that should never happen
- Roles or ownership transfers to the EOA or null address
- Self-distruct of the protocol contract
- Changes in immutable slots values
- ...

---

For more information, visit Forta and LIDO.
Let’s go **deeper**!

Practical examples
You should start thinking about alerts and analyzing code before the deployment.

**When we start?**

ADR: Review contracts architecture for early issues detection.

Develop: Add stuff necessary for proper alerting. Events, view methods, etc.

Review: Check that we have all we need for monitoring and alerting.

Deploy: Develop and deploy detection bots. Set-up alerting channels.
## Defining critical events

<table>
<thead>
<tr>
<th>ACL changes</th>
<th>Ownership transfers</th>
<th>State changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each ACL change should be alerted</td>
<td>To EOA</td>
<td>Ordinary state changes</td>
</tr>
<tr>
<td>Granting or revoking critical permissions should be supplied with the critical alerts</td>
<td>To unknown contract</td>
<td>Huge changes in ordinary state</td>
</tr>
<tr>
<td></td>
<td>To Null address</td>
<td>Critical state changes</td>
</tr>
</tbody>
</table>

Most common ACL contracts

- `@openzeppelin/contracts/access/AccessControl.sol`
- `@openzeppelin/contracts/access/Ownable.sol`
Defining protocol invariants

- Amount minted = Amount deposited
- Source bridge balance $\geq$ Target bridge balance
- Collateral value $>$ Loan value
- ...
### Defining repetitive events and ways to predict issues with it

<table>
<thead>
<tr>
<th>Repetitive Event</th>
<th>Possible issues</th>
<th>How to predict</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle report</td>
<td>Quorum not reached</td>
<td>Monitor quorum participation and difference in the reports</td>
</tr>
<tr>
<td>Rewards distribution</td>
<td>Rewards are not distributed in time</td>
<td>Off chain executor monitoring</td>
</tr>
<tr>
<td>Stake deposits</td>
<td>Huge amount of funds in buffer</td>
<td>Off chain executor monitoring</td>
</tr>
<tr>
<td>Validator keys upload</td>
<td>New keys are not uploaded</td>
<td>Monitor current available keys number</td>
</tr>
</tbody>
</table>
A good way to protect your contracts from being hacked is to investigate known hacks and make sure none of them is applicable for your code.
All alerts should be acted
Otherwise, they shouldn’t exist at all
Stay up-to date with the alerts
Set-up on-call system for critical alerts if possible

PagerDuty

Opsgenie
Stay up-to date with the alerts
Use separate chats for the info feed and critical alerts

Lido Mainnet Alerts: [HIGH] Significant Balancer Pool size ...

Lido Mainnet Alerts: [INFO] EasyTrack: New motion create...

Critical and High

All feed
Actions on **alerts**

**RunBooks**

RunBook is a comprehensive description of the alert itself and actions to be taken on it.

- **Description**
- **Severity**
- **Confirmation**
- **Resolution**
- **Escalation**
- **Notes and links**

On-call person or person on-duty should know what actions should be done when the alert fires.
Check out Lido RunBook
cutt.ly/4BtVNBq
Actions on alerts

Emergency brakes

@openzeppelin/contracts/security/Pausable.sol

function deposit() external payable whenNotPaused

Your contracts should have “emergency brakes”
Detecting hacks without ability to stop it is useless
Alerting for the most critical stuff should be duplicated
It is time to create your own alerts!
Protocol Alerts Exercises

**Exercise 1**
Operational monitoring and alerts

- Define main operational aspects of the protocol
- Define repetitive events
- Describe events ABI, alert texts and severity
- Think about protocol specific operations that you need to be alerted about

**Exercise 2**
Security monitoring and alerts

- Define critical events and state changes in the protocol
- Define ACL model
- Determine protocol invariants
- Implement alerts for all points above

Get started at: