Teatime Documentation

Release 0.3.1

Dominik Muhs

Mar 01, 2022

Contents:

CHAPTER 1

Teatime - A Blockchain RPC Attack Framework

1.1 Deployed a node? Have a cup.

Teatime is an RPC attack framework aimed at making it easy to spot misconfigurations in blockchain nodes. It detects a large variety of issues, ranging from information leaks to open accounts, and configuration manipulation.

The goal is to enable tools scanning for vulnerable nodes and minimizing the risk of node-based attacks due to common vulnerabilities. Teatime uses a plugin-based architecture, so extending the library with your own checks is straightforward.

Please note that this library is still a PoC and lacks documentation. If there are plugins you would like to see, feel free to contact me on Twitter!

1.2 Installation

Teatime runs on Python 3.6+.

To get started, simply run

```
$ pip3 install teatime
```

Alternatively, clone the repository and run

```
$ pip3 install .
```

Or directly through Python's `setuptools`:

```
$ python3 setup.py install
```

1.3 Example

To get started, simply instantiate a `Scanner` class and pass in the target IP, port, node type, and a list of instantiated plugins. Consider the following sample to check whether a node is synced and mining:

```
from teatime.scanner import Scanner
from teatime.plugins.context import NodeType
from teatime.plugins.eth1 import NodeSync, MiningStatus

TARGET_IP = "127.0.0.1"
TARGET_PORT = 8545
INFURA_URL = "Infura API Endpoint"

def get_scanner():
    return Scanner(
        ip=TARGET_IP,
        port=TARGET_PORT,
        node_type=NodeType.GETH,
        plugins=[
            NodeSync(infura_url=INFURA_URL, block_threshold=10),
            MiningStatus(should_mine=False)
        ]
    )

if __name__ == '__main__':
    scanner = get_scanner()
    report = scanner.run()
    print(report.to_dict())
```

Check out the examples directory for more small samples! Teatime is fully typed, so also feel free to explore options in your IDE if reading the documentation is not your preferred choice. :)

1.4 Future Development

The future of Teatime is uncertain, even though I would love to add broader checks that go beyond RPC interfaces, specifically for technologies such as:

- Ethereum 2.0
- Filecoin
- IPFS

If you want to integrate plugins for smaller, less meaningful chains such as Bitcoin or Ethereum knock-offs, feel free to fork the project and integrate them separately.

CHAPTER 2

Installation

2.1 Stable release

To install Teatime, run this command in your terminal:

```
$ pip install teatime
```

This is the preferred method to install Teatime, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Teatime can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/dmuhs/teatime
```

Or download the tarball:

```
$ curl -OJL https://github.com/dmuhs/teatime/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use Teatime in a project:

```
import teatime
```

CHAPTER 4

The Architecture of Teatime

In this document we will get to know the design decisions behind Teatime and introduce the library's primitives to make development more intuitive. There are five components to Teatime that make up its inner workings:

- **Scanner**: A scanner executes one or more plugins on a given target
- **Plugin**: Plugins are where the magic happens: They perform the checks and generate issues
- **Context**: The context object is passed from one scanner to the next and holds report meta data
- **Report**: The report object inside the context is where Plugins add their issues
- **Issue**: This basic building block holds data about findings such as title, severity, and much more

4.1 Scanner

The `Scanner` class is a user's entry point to Teatime. It takes an IP, port, node type, and a list of `Plugin` instances to execute on the target. Behind the scenes, it initializes a fresh `Context` class, which is passed between the plugins to aggregate report data.

In the current implementation, the `Scanner` class executes the given plugin list sequentially on the target. The list order is equivalent to the execution order. This means that a plugin could provide meta data in a report that another plugin further down the line can use. While this is not recommended because it introduces implicit dependencies, it can certainly be used to build highly customized and complex scanning pipelines.

After the scan is done, the `Scanner` class attaches the time elapsed for the scan to the report's meta data.

4.2 Plugin

The `Plugin` class is a base for all concrete scans of Teatime. A Plugin can execute one or more checks on the given target. To specify your own behaviour, the base `Plugin` class contains an abstract method `_check` that can be overridden by the user. This method gets a `Context` object (hopefully) containing all relevant meta data needed by the plugin.

For RPC interaction specifically, the `Plugin` class contains a helper method to query RPC endpoints in a robust way and handle connection errors along the way. To prevent Teatime from crashing completely in case of plugin-related errors, it is recommended to raise and reraise a `PluginException`. This can be caught easily on the top level, e.g. by the routine executing the scanner.

4.3 Context

The `Context` object contains report- and target-related information instructing the `Plugin` classes. It contains the target, the `Report` object, node type, and an extra data dictionary for any additional information that e.g. needs to be passed further down the plugin pipeline. Per scan, there is only one `Context` instance (initialized in the `Scanner` at the beginning of a scan), which gets shared across Plugin instances as they are executed in the pipeline.

4.4 Report

The `Report` object is essentially a container for `Issue` objects, along with meta data on the executed scan. It contains a UUID for traceability, the target, a creation timestamp, a list of issues, and a meta data dictionary for any additional information that should be communicated to the user.

The idea of wrapping issues in a report and duplicating information such as the target is that each `Report` object should be independent. Thinking of developers who might want to pass Teatime report data into a database, or export it as a file, this is a nice property to have, because no additional data apart from the object itself is required.

Furthermore, the `Report` class contains various helper methods that make serialization and common checks, such as checking for high-severity issues, easier.

4.5 Issue

This is the lowest-level primitive. The `Issue` object can be added to a `Report` instance by using its `add_issue` method. Each issue contains a UUID for traceability, a title, description, severity, and a raw data field. The latter one is meant to contain the raw RPC response. For example, if a scan has detected an information leak, the raw data field can be populated with the actual leaking information text, without cluttering the title or description - thus keeping issues readable and allowing the user to omit large strings when presenting an issue to the user.

Just like the `Report` class, an `Issue` object contains various helper method for easier serialization, as well as determining whether the issue at hand is severe. Wrapping issues into their own object has the advantage of enforcing a standard format across plugins and requiring them to provide information that help the user make sense of what has been reported.

CHAPTER 5

teatime

5.1 teatime package

5.1.1 Subpackages

teatime.plugins package

Subpackages

teatime.plugins.eth1 package

Submodules

teatime.plugins.eth1.account_creation module

This module contains a plugin with checks for account creation.

class teatime.plugins.eth1.account_creation.**AccountCreation**(*test_password: str*)

Bases: *teatime.plugins.base.JSONRPCPlugin*

Detect whether it's possible to create an account on the node.

Severity: Medium

This check will try to generate a new account on the node using the `personal_newAccount` and lock the new account with the given password.

Geth: https://geth.ethereum.org/docs/rpc/ns-personal#personal_newaccount Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-personal-module#personal_newaccount

INTRUSIVE = True

teatime.plugins.eth1.account_import module

This module holds the plugin checking for accound imports.

class teatime.plugins.eth1.account_import.**GethAccountImport**(*keydata: str*, *password: str*)

Bases: *teatime.plugins.base.JSONRPCPlugin*

Detect whether it's possible to import an account on the node.

Severity: Medium

This check will try to import an existing account on the node using the `personal_importRawKey` and lock the new account with the given password. This check only works with Geth client nodes.

Geth: https://geth.ethereum.org/docs/rpc/ns-personal#personal_importrawkey

INTRUSIVE = True

teatime.plugins.eth1.gas_limits module

This module contains plugins around the gas-setting RPC endpoints.

class teatime.plugins.eth1.gas_limits.**ParityGasCeiling**(*gas_target: int*)

Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to set a new gas ceiling target for mined blocks.

Severity: Critical

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity_set-module#parity_setgasceiltarget

INTRUSIVE = True

class teatime.plugins.eth1.gas_limits.**ParityGasFloor**(*gas_floor: int*)

Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to set a new gas floor target for mined blocks.

Severity: Critical

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity_set-module#parity_setgasfloortarget

INTRUSIVE = True

teatime.plugins.eth1.information_leaks module

This module contains plugins with admin interface checks.

class teatime.plugins.eth1.information_leaks.**GethDatadir**

Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to fetch Geth's data directory.

Severity: Low

Geth: https://geth.ethereum.org/docs/rpc/ns-admin#admin_datadir

INTRUSIVE = False

class teatime.plugins.eth1.information_leaks.**GethNodeInfo**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to fetch admin info about the node.

Severity: Low

Geth: https://geth.ethereum.org/docs/rpc/ns-admin#admin_nodeinfo

INTRUSIVE = False

class teatime.plugins.eth1.information_leaks.**ParityDevLogs**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to fetch the node's developer logs.

Severity: Critical

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity-module#parity_devlogs

INTRUSIVE = False

class teatime.plugins.eth1.information_leaks.**PeerlistLeak**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to fetch peer list information.

Severity: Medium

Geth: https://geth.ethereum.org/docs/rpc/ns-admin#admin_peers Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity-module#parity_netpeers

INTRUSIVE = False

teatime.plugins.eth1.manipulation module

This module contains plugins around setting vital execution parameters.

class teatime.plugins.eth1.manipulation.**ParityChangeCoinbase**(*author: str*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to change the coinbase address.

Severity: Critical

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity_set-module#parity_setauthor

INTRUSIVE = True

class teatime.plugins.eth1.manipulation.**ParityChangeExtra**(*extra_data: str*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to set the extra data field.

Severity: Low

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity_set-module#parity_setextradata

INTRUSIVE = True

class teatime.plugins.eth1.manipulation.**ParityChangeTarget**(*target_chain: str*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to change the target chain.

Severity: Critical

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity_set-module#parity_setchain

INTRUSIVE = True

class teatime.plugins.eth1.manipulation.**ParitySyncMode**(*mode: str*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to set the node's sync mode.

Severity: Critical

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity_set-module#parity_setmode

INTRUSIVE = True

teatime.plugins.eth1.mining module

This module contains a plugin for mining-related checks.

class teatime.plugins.eth1.mining.**HashrateStatus**(*expected_hashrate: int*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Check whether the node has a certain hash rate.

Severity: Medium

This plugin will use the `eth_hashrate` method to fetch the node's hash rate. If the hash rate is different from a user-specified value, an issue will be logged.

INTRUSIVE = False

class teatime.plugins.eth1.mining.**MiningStatus**(*should_mine: bool*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Check whether the node is mining.

Severity: Medium

This plugin will use the `eth_mining` method to find out whether a node is mining or not. If there is a difference to the user-specified value, an issue will be logged.

INTRUSIVE = False

teatime.plugins.eth1.network module

This module contains a plugin for network-related checks.

class teatime.plugins.eth1.network.**NetworkListening**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Check whether the node is listening for peers.

Severity: High

This plugin will use the `net_listening` method to check whether the node is listening to new peers. If that is not the case, an issue will be logged.

INTRUSIVE = False

class teatime.plugins.eth1.network.**ParityDropPeers**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to remove non-reserved peers from the peer list.

Severity: Critical

This plugin will attempt to drop all non-reserved peer entries from the node's peer table.

INTRUSIVE = True

class teatime.plugins.eth1.network.**PeerCountStatus**(*minimum_peercount: int*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Check whether the node has a certain peer count.

Severity: Medium

This plugin will use the `net_peerCount` method to check the node's peer count. If the value is lower than the user-specified value of minimum peers, an issue will be logged.

INTRUSIVE = False

class teatime.plugins.eth1.network.**PeerlistManipulation**(*test_enode: str*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to add a peer to the node's peer list.

Severity: High

This plugin will attempt to add a given peer to the node's peer list.

INTRUSIVE = True

teatime.plugins.eth1.open_accounts module

This module contains a plugin checking for account-related issues.

class teatime.plugins.eth1.open_accounts.**AccountUnlock**(*infura_url: str*, *wordlist=None*, *skip_below: int = None*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Check whether any accounts on the node are weakly protected.

Severity: Critical

This plugin will use the `eth_accounts` method to find accounts registered on the target node, and attempt to unlock the accounts with a given set of passwords. Each account is unlocked for a time of one second, the minimum time possible.

Optionally, accounts below a minimum balance can be skipped.

INTRUSIVE = True

class teatime.plugins.eth1.open_accounts.**OpenAccounts**(*infura_url: str*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Check for any accounts registered on the node.

Severity: Medium

This plugin will use the `eth_accounts` method to find accounts registered on the target node, and fetch the account's latest balance through Infura.

INTRUSIVE = False

teatime.plugins.eth1.rpc module

This module contains plugins for controlling the HTTP RPC server status.

class teatime.plugins.eth1.rpc.**GethStartRPC**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to start the RPC service.

Severity: Critical

This plugin attempts to start the HTTP RPC interface using the `admin_startRPC` method.

INTRUSIVE = True

class teatime.plugins.eth1.rpc.**GethStopRPC**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to stop the RPC service.

Severity: Critical

Talking about shooting yourself in the foot. This plugin attempts to stop the HTTP RPC interface using the `admin_stopRPC` method. In case you didn't notice, this might affect the outcome of other plugins due to connection failures.

INTRUSIVE = True

teatime.plugins.eth1.sha3 module

This module contains a plugin to check for SHA3 consistency.

class teatime.plugins.eth1.sha3.**SHA3Consistency**(*test_input: str*, *test_output: str*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Check for SHA3 consistency.

Severity: Critical

This plugin submits a user-specified value and lets the node convert it into a SHA3 hash using the `web3_sha3` method. If the result value is different from the user-specified output value, an issue is logged.

INTRUSIVE = False

teatime.plugins.eth1.sync module

This module contains a plugin checking for node sync issues.

class teatime.plugins.eth1.sync.**NodeSync**(*infura_url*, *block_threshold: int = 10*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Check the node's sync state and whether it's stuck.

Severity: None/Critical

This plugin fetches the sync state if the node. If it is not syncing, the most recent block number is fetched from Infura using the `eth_blockNumber` method. If the most recent block number is higher than the node's block number with a certain threshold, the node might be stuck and out of sync with the mainnet. In that case, a critical issue is logged. Otherwise, an informational issue on the current sync state is logged.

INTRUSIVE = False

teatime.plugins.eth1.tx_limits module

This module contains plugins around setting transaction-related limits.

class teatime.plugins.eth1.tx_limits.**ParityMinGasPrice**(*gas_price: int*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to set the minimum transaction gas price.

Severity: Critical

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity_set-module#parity_setmingasprice

INTRUSIVE = True

class teatime.plugins.eth1.tx_limits.**ParityTxCeiling**(*gas_limit: int*)
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to set the maximum transaction gas.

Severity: Critical

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity_set-module#parity_setmaxtransactiongas

INTRUSIVE = True

teatime.plugins.eth1.txpool module

This module contains checks regarding a node's transaction pool.

class teatime.plugins.eth1.txpool.**GethTxPoolInspection**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to inspect the transaction pool.

Severity: Low

Geth: https://geth.ethereum.org/docs/rpc/ns-txpool#txpool_inspect

INTRUSIVE = False

class teatime.plugins.eth1.txpool.**GethTxPoolStatus**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to fetch the transaction pool status.

Severity: Low

Geth: https://geth.ethereum.org/docs/rpc/ns-txpool#txpool_status

INTRUSIVE = False

class teatime.plugins.eth1.txpool.**ParityTxPoolStatistics**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to fetch the transaction pool statistics.

Severity: Low

Parity: https://openethereum.github.io/wiki/JSONRPC-parity-module#parity_pendingtransactionsstats

INTRUSIVE = False

class teatime.plugins.eth1.txpool.**TxPoolContent**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to fetch the transaction pool contents.

Severity: Low

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity-module#parity_pendingtransactions Geth: https://geth.ethereum.org/docs/rpc/ns-txpool#txpool_content

INTRUSIVE = False

teatime.plugins.eth1.upgrade module

This module contains a plugin checking for Parity/OpenEthereum upgrades.

class teatime.plugins.eth1.upgrade.**ParityUpgrade**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to check for an available upgrade.

Severity: Critical

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-parity_set-module.html#parity_upgradeready

INTRUSIVE = False

teatime.plugins.eth1.version module

This module contains a plugin to check for stale node versions.

class teatime.plugins.eth1.version.**NodeVersion**(*geth_url: str = 'https://api.github.com/repos/ethereum/go-ethereum/releases/latest'*, *parity_url: str = 'https://api.github.com/repos/openethereum/openethereum/releas*
Bases: *teatime.plugins.base.JSONRPCPlugin*

Check whether a given node's version is stale.

Severity: None/High

This plugin will fetch the client's version string, and attempt to extract the node's semantic version number. For Geth and Parity/OpenEthereum, it will try to fetch the latest repository tag and compare both versions. If there is a mismatch, an issue is logged about the node version being stale. In any case, an informational issue will be logged containing the version string.

Parity/OpenEthereum: https://openethereum.github.io/wiki/JSONRPC-web3-module#web3_clientversion Geth: I couldn't find the web3 namespace in the official docs :(

INTRUSIVE = False

static latest_repo_release(*url: str*) → str
Fetch the latest release tag for the given repository URL.

This method will use the public Github API to fetch the latest release tag for the given repository.

Returns The repo's semver as a string

teatime.plugins.eth1.websocket module

This module contains plugins for controlling the Websocket RPC server status.

class teatime.plugins.eth1.websocket.**GethStartWebsocket**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to start the websocket service.

Severity: Critical

Geth: https://geth.ethereum.org/docs/rpc/ns-admin#admin_startws

INTRUSIVE = True

class teatime.plugins.eth1.websocket.**GethStopWebsocket**
Bases: *teatime.plugins.base.JSONRPCPlugin*

Try to stop the websocket service.

Severity: Critical

Geth: https://geth.ethereum.org/docs/rpc/ns-admin#admin_stopws

INTRUSIVE = True

Module contents

This package contains plugins related to Ethereum 1.0.

teatime.plugins.ipfs package

Submodules

teatime.plugins.ipfs.add module

This module contains plugins regarding file uploads to the node.

class teatime.plugins.ipfs.add.**OpenUploadAdd**(*file_name: str = '.teatime'*, *file_content: str = 'teatime test file'*)
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Detect where it's possible to upload a file using the /add endpoint.

Severity: High

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-add

An open upload functionality can enable an attacker to upload a lot of random data until storage space is exhausted, thus performing a denial of service attack against future uploads.

INTRUSIVE = True

class teatime.plugins.ipfs.add.**OpenUploadTarAdd**(*file_name: str = '.teatime'*, *file_content: str = 'teatime test file'*)
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Detect where it's possible to upload a file using the /tar/add endpoint.

Severity: High

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-tar-add

An open upload functionality can enable an attacker to upload a lot of random data until storage space is exhausted, thus performing a denial of service attack against future uploads.

INTRUSIVE = True

teatime.plugins.ipfs.commands module

This module contains plugins regarding commands surfaced by the node.

class teatime.plugins.ipfs.commands.**CommandCheck**(*allowlist: Optional[Sequence[Sequence[str]]] = None*, *denylist: Optional[Sequence[Sequence[str]]] = None*)

Bases: *teatime.plugins.base.IPFSRPCPlugin*

Detect whether disallowed commands are enabled.

Severity: High

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-commands

The IPFS API offers a lot of endpoints, some of which might be accidentally enabled. This plugin attempts to fetch the list of enabled API commands and will log an issue of user-specified commands are enabled, or not enabled.

INTRUSIVE = False

teatime.plugins.ipfs.files module

This module contains plugins regarding listing files provided by the node.

class teatime.plugins.ipfs.files.**CIDFSEnum**(*cid_paths: Sequence[str] = None*)

Bases: *teatime.plugins.base.IPFSRPCPlugin*

Check whether the given CIDs are present on the node.

Severity: Medium

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-ls Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-file-ls

A common IPFS file path is leaking directory contents of UNIX filesystem objects. Depending on where IPFS has been mounted, this can leak f"confidential information.

INTRUSIVE = False

check_paths(*context: teatime.plugins.context.Context*, *endpoint: str*)

class teatime.plugins.ipfs.files.**FilestoreEnum**

Bases: *teatime.plugins.base.IPFSRPCPlugin*

Check whether the objects in the filestore can be listed.

Severity: Medium

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-filestore-ls

The filestore endpoint is leaking contents of its objects. An attacker can use this endpoint to enumerate potentially confidential data on the system.

INTRUSIVE = False

class teatime.plugins.ipfs.files.**UnixFSEnum**(*path: str = None*)
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Check whether the objects in the local mutable namespace can be listed.

Severity: Medium

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-files-ls

The UNIX root directory path is leaking contents of UNIX filesystem objects. An attacker can use this endpoint along with the /files/read endpoint to enumerate potentially confidential data on the system.

INTRUSIVE = False

teatime.plugins.ipfs.keys module

This module contains plugins regarding listing and extracting keys.

class teatime.plugins.ipfs.keys.**KeyLeaks**(*export: bool = False*)
Bases: *teatime.plugins.base.IPFSRPCPlugin*

List and attempt to export the node's keys.

Severity: CRITICAL

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-key-export

The version endpoint reveals the Go version IPFS has been compiled with, along with repository and system information, which may contain sensitive data.

INTRUSIVE = False

teatime.plugins.ipfs.logs module

This module contains plugins regarding log information leaked by the node.

class teatime.plugins.ipfs.logs.**ChangeLogLevel**(*subsystem: str = 'all', level: str = 'info'*)
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to change the log level for the given subsystems.

Severity: Medium

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-log-level

Anyone can change the log level of messages generated by the node. Log messages, especially debug-level ones, can leak sensitive information about the node's setup and operations running on it. An attacker may unlock additional information by enabling debug logs. This could also results in degraded performance, espeically when logs are stored in local files, or in log aggregation systems unable to handle the load.

INTRUSIVE = True

class teatime.plugins.ipfs.logs.**EnumerateLogs**
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to list all logging subsystems.

Severity: Low

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-log-ls

It is possible to list the logging subsystems that the node is using. This may be used by an attacker to find non-standard customizations on the node, as well as fingerprint the node setup for identification.

INTRUSIVE = False

class teatime.plugins.ipfs.logs.**ReadLogs**(*line_limit: int = 1*)
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Gather a sample of log data from the node's subsystems.

Severity: Medium

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-log-tail

Anyone can list log messages generated by the node. Log messages, especially debug-level ones, can leak sensitive information about the node's setup and operations running on it.

INTRUSIVE = False

teatime.plugins.ipfs.p2p module

This module contains plugins regarding leaked P2P network information

class teatime.plugins.ipfs.p2p.**P2PCloseStream**
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to close all active P2P streams.

Severity: High

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-p2p-stream-close

Anyone is able to close active P2P streams on this node. This exposed functionality may be used by an attacker to disrupt the node's availability and block connections.

INTRUSIVE = True

class teatime.plugins.ipfs.p2p.**P2PCreateListener**
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to enable forwarding new connections to the libp2p service.

Severity: High

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-p2p-listen

Anyone is able to register P2P listeners on this node. This exposed functionality may be used by an attacker to disrupt the node's availability and block connections.

INTRUSIVE = True

class teatime.plugins.ipfs.p2p.**P2PEnableForwarding**
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to enable forwarding new connections to the libp2p service.

Severity: High

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-p2p-forward

Anyone is able to register P2P forwardings on this node. This exposed functionality may be used by an attacker to disrupt the node's availability and block connections.

INTRUSIVE = True

class teatime.plugins.ipfs.p2p.**P2PListListeners**
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to list all active P2P listeners.

Severity: Low

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-p2p-ls

Anyone is able to list the P2P listener services running on this node. This method may leak internal information on other peer-to-peer services running on this node.

INTRUSIVE = False

class teatime.plugins.ipfs.p2p.**P2PListStreams**
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to list all active P2P streams.

Severity: Low

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-p2p-stream-ls

Anyone is able to list the active P2P streams on this node. This method may leak internal information on other peer-to-peer services and connections on this node.

INTRUSIVE = False

class teatime.plugins.ipfs.p2p.**P2PStopForwarding**
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to stop the node from listening to new connection forwards.

Severity: High

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-p2p-close

Anyone is able to close active P2P forwardings on this node. This exposed functionality may be used by an attacker to disrupt the node's availability and block connections.

INTRUSIVE = True

teatime.plugins.ipfs.pins module

This module contains plugins regarding listing and manipulating a node's pins.

class teatime.plugins.ipfs.pins.**AddPin**(*cid: str = 'Qmf9vKuR6MnTEGYXhzwpMib5EFGoXPWCJh3mXTvasb3Cas'*)
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Detect where it's possible to add new pin.

Severity: High

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-pin-add

Open pinning can enable an attacker to flush a large amount of random data onto the node's disk until storage space is exhausted, thus performing a denial of service attack against future uploads/pins.

INTRUSIVE = True

class teatime.plugins.ipfs.pins.**EnumeratePins**(*cid: str = 'Qmf9vKuR6MnTEGYXhzwpMib5EFGoXPWCJh3mXTvasb3Cas'*)
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Detect where it's possible to list the node's pins.

Severity: Low

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-pin-ls

It is possible to list all the content IDs that are pinned to the node's local storage.

INTRUSIVE = False

class teatime.plugins.ipfs.pins.**RemovePin**(*pin: str = 'Qmf9vKuR6MnTEGYXhzwpMib5EFGoXPWCJh3mXTvasb3C restore: bool = True*)

Bases: *teatime.plugins.base.IPFSRPCPlugin*

Detect where it's possible to remove the node's pins.

Severity: High

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-pin-rm

It is possible to remove all the content IDs that are pinned to the node's local storage. This poses a risk to data availability as an attacker can unpin any file.

INTRUSIVE = True

teatime.plugins.ipfs.shutdown module

This module contains a plugin to test a node's remote shutdown functionality.

class teatime.plugins.ipfs.shutdown.**Shutdown**

Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to list all active P2P listeners.

Severity: Critical

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-shutdown

Anyone can shut down the IPFS daemon. This plugin has shut down the node. This is the highest possible threat to availability. Why would you leave this enabled? Are you insane?

INTRUSIVE = True

teatime.plugins.ipfs.version module

This module contains plugins to probe a node's version and find outdated dependencies.

class teatime.plugins.ipfs.version.**DependencyVersion**(*check_dependencies: bool = True*)

Bases: *teatime.plugins.base.IPFSRPCPlugin*

Detect whether the node's version endpoint is available.

Severity: Low

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-version-deps

The version endpoint reveals the Go version IPFS has been compiled with, along with repository and system information, which may contain sensitive data.

INTRUSIVE = False

class teatime.plugins.ipfs.version.**Version**

Bases: *teatime.plugins.base.IPFSRPCPlugin*

Detect whether the node's version endpoint is available.

Severity: Low

Endpoint: https://docs.ipfs.io/reference/http/api/#api-v0-version

The version endpoint reveals the Go version IPFS has been compiled with, along with repository and system information, which may contain sensitive data.

INTRUSIVE = False

teatime.plugins.ipfs.webui module

This module contains a plugin detect a node's exposed web interface.

class teatime.plugins.ipfs.webui.**WebUIEnabled**(*route: str = '/webui'*)
Bases: *teatime.plugins.base.IPFSRPCPlugin*

Attempt to access the target's Web UI.

Severity: HIGH

Anyone can access the Web UI. A plethora of administrative actions can be done through the web interface. This includes changing the node's configuration, which can be used to open other potential attack vectors.

INTRUSIVE = False

static fetch_ui(*target*, *route*)

Module contents

Submodules

teatime.plugins.base module

This module holds the base plugin class and exception.

class teatime.plugins.base.**BasePlugin**
Bases: abc.ABC

The base plugin class.

INTRUSIVE = True

run(*context: teatime.plugins.context.Context*)
The plugin's entrypoint as invoked by the scanner.

This method will call the plugin's `_check` method, which should be overridden by concrete JSONRPC-Plugin instances. It will catch any `PluginException` and skip the execution. In any case, at the end of the check run, the plugin name is added as a meta field to denote that it has been executed.

Parameters **context** – The context object containing report-related information

class teatime.plugins.base.**IPFSRPCPlugin**
Bases: *teatime.plugins.base.BasePlugin*, abc.ABC

static get_rpc_json(*target: str*, *route: str = ''*, *params: Union[dict, Sequence[tuple]] = None*, *headers: Optional[dict] = None*, *files: Optional[dict] = None*, *raw: bool = False*, *timeout: int = 3*, *stream_limit: int = None*)
Send a request to the IPFS HTTP API.

Parameters

- **target** – The target to send the request to
- **route** – The URL to send the API request to
- **params** – A dict of URL parameters to add
- **headers** – Optional headers to attach
- **files** – A dictionary of files to upload
- **raw** – If true, the result will not interpreted as JSON
- **timeout** – Number of seconds to wait until timing out
- **stream_limit** – Maximum number of lines to read

Returns

class teatime.plugins.base.**JSONRPCPlugin**

Bases: *teatime.plugins.base.BasePlugin*, abc.ABC

A base plugin for JSON-RPC APIs.

static get_rpc_int(*target*, *method*, *params: List[str] = None*, *idx: int = 1*) → int

Attempt to make an RPC call and decode the result as an integer.

Parameters

- **target** – The RPC target URL
- **method** – The RPC method
- **params** – Additional RPC method params (optional)
- **idx** – RPC call index (optional)

Returns The payload result as integer

Raises ***PluginException*** – If connection or payload-related errors occur

static get_rpc_json(*target: str*, *method: str*, *params: List[Union[str, int]] = None*, *idx: int = 0*)

Execute an RPC call against a given target.

The current timeout for the RPC request is three seconds. Any `PluginException` instances raised, contain the reason in the message string, e.g. if a connection failure occurred, the response status code was not 200, an error field is present, or if the result field is left empty.

Parameters

- **target** – The target URI to send the request to
- **method** – The RPC method to use
- **params** – Additional parameters for the method (optional)
- **idx** – The RPC call's ID (optional)

Returns The response payload's "result" field

Raises ***PluginException*** – If the request faied or the response is inconsistent

exception teatime.plugins.base.**PluginException**

Bases: Exception

An exception for plugin-related errors.

teatime.plugins.base.**handle_connection_errors**(*func*)
Catch connection-related excetions and reraise.

This slim wrapper will catch connection and decoding errors, and requests-related exceptions to reraise them as PluginErrors so we can catch them in a standardized way.

teatime.plugins.context module

This module contains the context that is passed to plugins.

class teatime.plugins.context.**Context**(*target*, *report*, *node_type*, ***kwargs*)
Bases: object

The context object passed between plugins.

class teatime.plugins.context.**NodeType**
Bases: enum.Enum

An Enum denoting a node type to scan.

Currently, only Geth and Parity/OpenEthereum are supported. Future considerations are: - IPFS - Filecoin - ETH2 clients

GETH = 0

IPFS = 2

PARITY = 1

Module contents

The package holding all Teatime plugins.

teatime.reporting package

Submodules

teatime.reporting.issue module

This module contains data structures regarding issues.

class teatime.reporting.issue.**Issue**(*uuid: str = None*, *title: str = None*, *description: str = None*, *severity: teatime.reporting.issue.Severity = None*, *raw_data: Any = None*)
Bases: object

An object describing a vulnerability, weakness, or informational message.

is_complete() → bool
Returns whether the issue is complete.

Returns A boolean indicating that the issue is complete

is_severe() → bool
Returns whether the issue is considered severe.

Returns A boolean indicating whether the issue is severe

to_dict() → dict
Converts the issue instance into a Python dict.

Returns A dict representing the issue

class teatime.reporting.issue.**Severity**
Bases: enum.Enum

An Enum denoting the severities an issue can have.

CRITICAL = 4

HIGH = 3

LOW = 1

MEDIUM = 2

NONE = 0

teatime.reporting.report module

This module contains the reporting functionality.

class teatime.reporting.report.**Report**(*target*, *uuid: str = None*, *issues=None*, *timestamp: str = None*)
Bases: object

A report class holding multiple issues and meta data.

add_issue(*issue: teatime.reporting.issue.Issue*)
Add an issue to the report.

Parameters **issue** – The issue object to add

add_meta(*key*, *value*)
Add a meta data key-value pair to the report.

Parameters

- **key** – The meta data key name
- **value** – The meta data key's value to attach

to_dict() → dict
Convert the report and its issues to a Python dict.

Returns The report's representation as a dict

Module contents

This package contains classes related to reports and issues.

teatime.scanner package

Submodules

teatime.scanner.scanner module

This module contains a scanner class running various Plugins.

class teatime.scanner.scanner.**Scanner**(*ip: str, port: int, node_type: teatime.plugins.context.NodeType, plugins: List[Union[teatime.plugins.base.JSONRPCPlugin, teatime.plugins.base.IPFSRPCPlugin]], prefix: str = 'http://'*)

Bases: object

The scanner class holding multiple plugins.

run() → teatime.reporting.report.Report
Run the scanner to generate a report.

Returns A report object holding all findings

Module contents

This package contains the scanner class.

5.1.2 Submodules

5.1.3 teatime.utils module

This module contains various utility functions around scanning.

teatime.utils.**check_port**(*host: str*, *port: int*, *timeout: int = 2*) → bool
Check whether a given port is available on the target host.

This helper function will attempt to connect to a given port on the target host.

Parameters

- **timeout** – Number of seconds to time out after
- **host** – The target host to connect to
- **port** – The target port to connect to

Returns A boolean indicating whether the connection was successful

teatime.utils.**reverse_dns**(*address: str*) → str
Attempt to resolve an IP address to its DNS name.

Parameters **address** – The IP address to resolve

Returns The IP's DNS name as a string

5.1.4 Module contents

Let's toast some nodes, eh?

CHAPTER 6

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/dmuhs/teatime/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

6.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

6.1.4 Write Documentation

Teatime could always use more documentation, whether as part of the official Teatime docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/dmuhs/teatime/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

6.2 Get Started!

Ready to contribute? Here's how to set up *teatime* for local development.

1. Fork the *teatime* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/teatime.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv teatime
$ cd teatime/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 teatime tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

6.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/dmuhs/teatime/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ pytest tests.test_teatime
```

6.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 7

Credits

7.1 Development Lead

- Dominik Muhs <dmuhs@protonmail.ch>

7.2 Contributors

None yet. Why not be the first?

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

T

U

V

W