5. Lightweight Authenticity and Integrity Monitoring of Devices in Operation

A
Creation & Confirmation

NEC Corporation

Verifying the software by monitoring the authenticity of the software in operation implemented on IoT devices with limited performance and memory capacity

Technical Features

■ Continuous monitoring of IoT devices（not limited to booting）

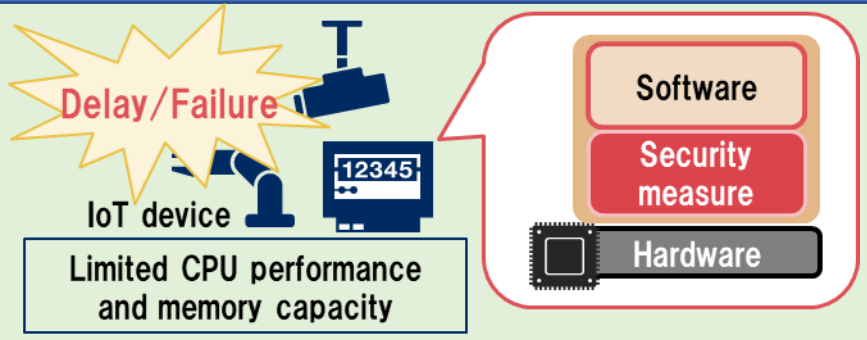
Improving the safety of IoT devices in continuous operation by monitoring authenticity and integrity of the execution codes and paths of IoT devices

■ Supporting introduction of the monitoring function by automatic development tool

Making it easy to introduce the monitoring function to software of target IoT device. Minimizing the development cost even when there is device or software update

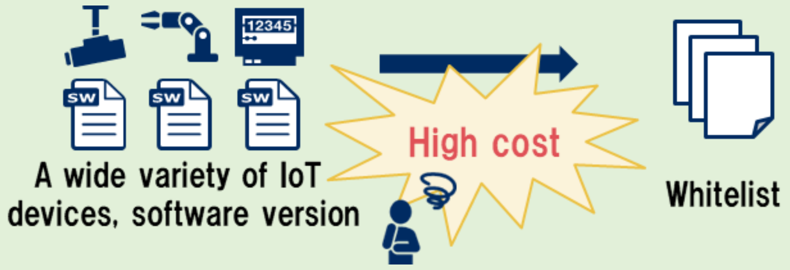
Problems of Introducing Security Measures to IoT Devices

Possibility of the impact on the operation



Most common attacks against software
- Code injection
- Control flow（execution path）injection

Deployment cost to a wide variety of devices



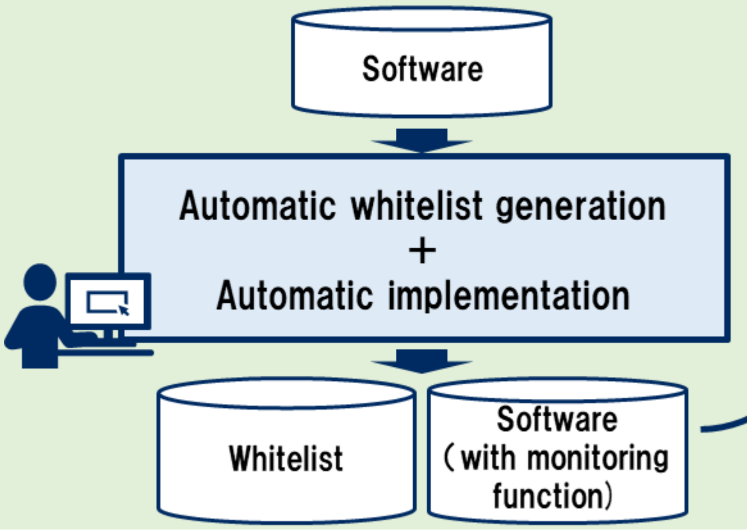
Existing technologies needs criteria（whitelist）for each target software
（based on hash values, processing structure, etc.）

Overview of R&D Technologies toward Solving Problems

Development phase

Whitelist Generation

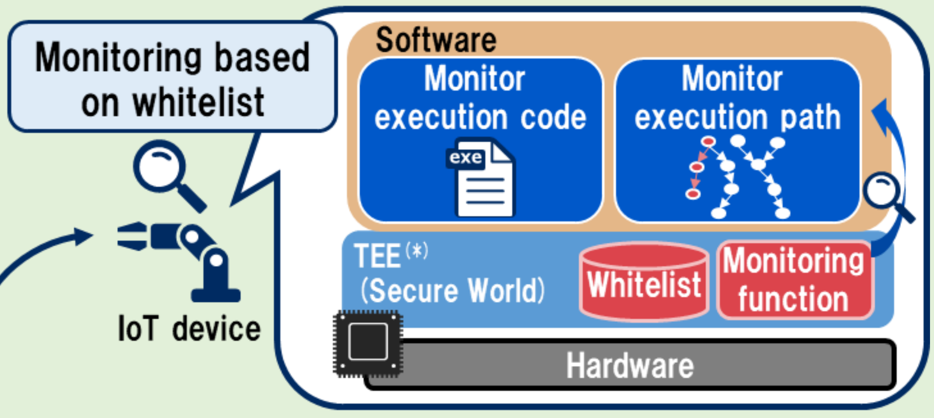
Generating whitelist automatically to reduce monitoring overhead, and implementing monitoring function into IoT devices automatically



Operation phase

Low-load Monitoring Function

Improving monitoring granularity by low-load monitoring both execution code and path



（*）Lightweight implementation utilizing hardware supported security extension, TEE (Trusted Execution Environment)

Technical Differences

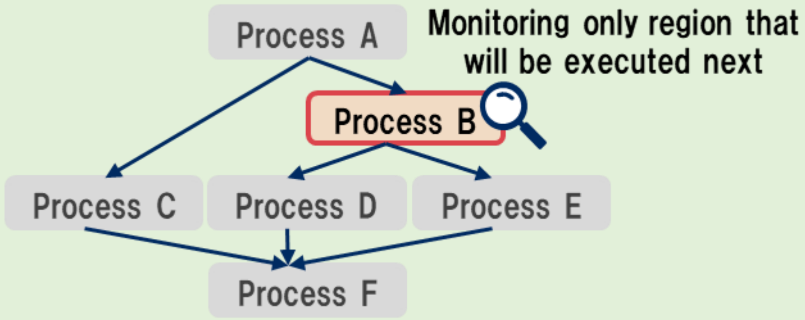
High Speed — Reduce overhead with software structure-based monitoring

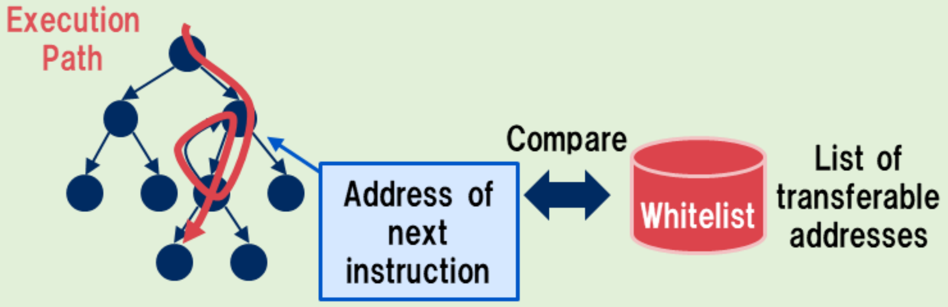
■ Monitoring execution code

- Existing methods monitor all/part of execution code periodically, which increases overhead
- Proposed method can reduce overhead by monitoring only the specific region based on software structure

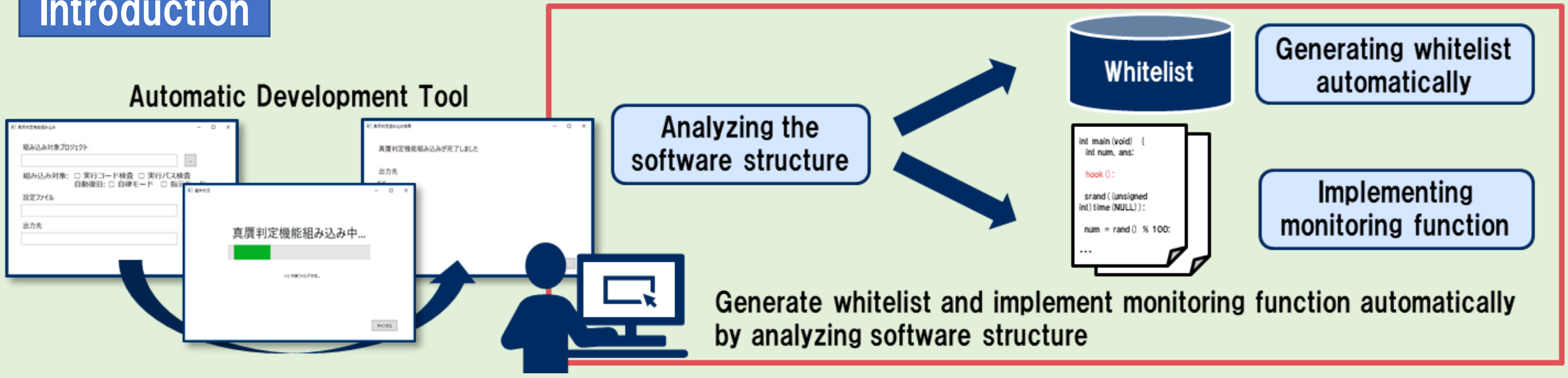
■ Monitoring execution path

- Existing methods requires a relatively heavy tasks (e.g. duplicating stack memory)
- Proposed method can reduce overhead by simple monitoring based on whitelist





Ease of Introduction — Implement monitoring function optimized for the software automatically


Automatic Development Tool
Analyzing the software structure
Whitelist
Generating whitelist automatically
Implementing monitoring function

組み込み対象プロジェクト

組み込み対象: □ 実行コード検査 □ 実行パス検査
自動復旧: □ 自律モード □ 指示モ

設定ファイル

出力先

真贋判定機能組み込み中...

真贋判定機能組み込みが完了しました

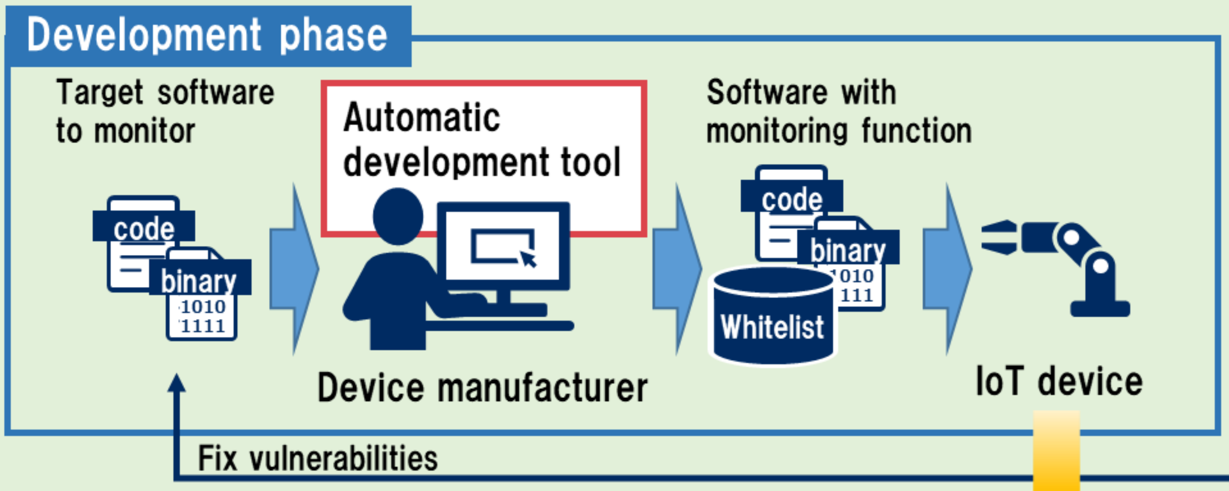
出力先

```
int main (void)   {
  int num, ans;

  hook () ;

  srand ( (unsigned
int) time (NULL) ) ;

  num  = rand ()  % 100;

...
```

Generate whitelist and implement monitoring function automatically by analyzing software structure

Use Cases

Support introduction into IoT devices used in the system of manufacturing, distribution, smart building, etc. and realize safety operation of IoT devices

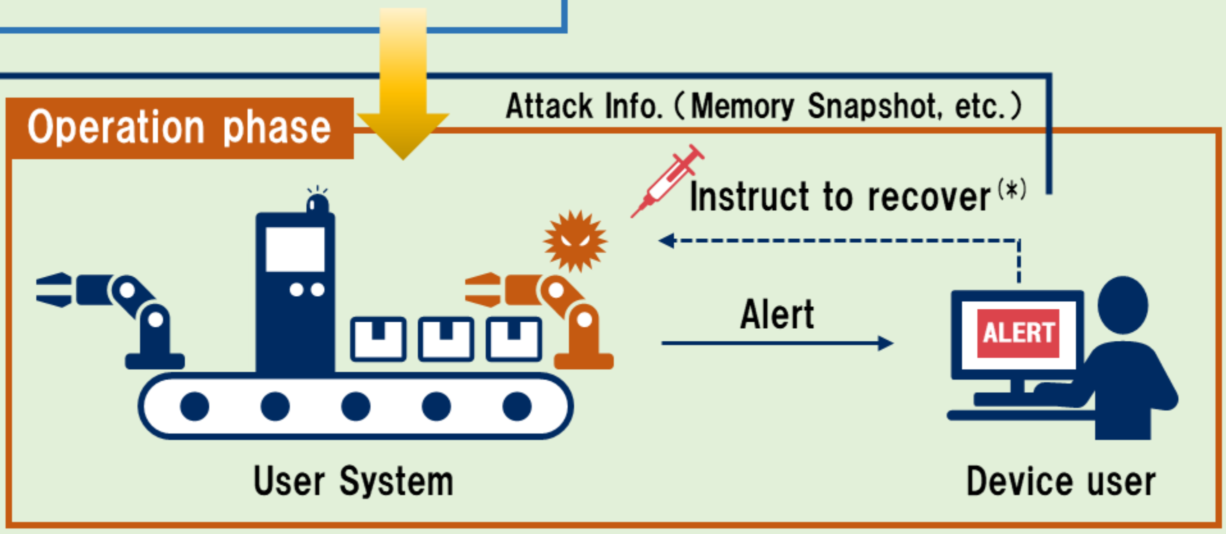


Device Manufacturer

- Support implementing the monitoring function into IoT device by automatic development tool
- Fix vulnerabilities based on attack info. from device users

Device User

- Receive an alert when the monitoring function detects tampering
- Check the safety of devices at any time by using IoT devices with the monitoring function



(*) Research and development of the automated recovery of tampered execution code is also in progress