Bookworm Game:
Automatic Discovery of LTE Vulnerabilities Through Documentation Analysis

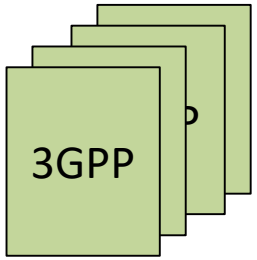
Yi Chen, Yepeng Yao, XiaoFeng Wang,Dandan Xu, Chang Yue,
Xiaozhong Liu, Kai Chen, Haixu Tang, Baoxu Liu

2021IEEE Symposium on Security and Privacy

Presenter: Wooyoung Go

Cellular network

- Cellular network is ubiquitous
- Cellular system is complicated
  - The 3GPP standard contains more than 3,235 specifications
  - Each spec is pretty long
- Due to this complexity, it has many security problem



3,235

https: onsiderations-m2m-deployment/

Vulnerability discovery in cellular networks

- There are many vulnerabilities in the cellular network
- Mostly found vulnerabilities in **ad-hoc & manual** analysis
- These approaches are costly, error-prone and often impossible

Introduction

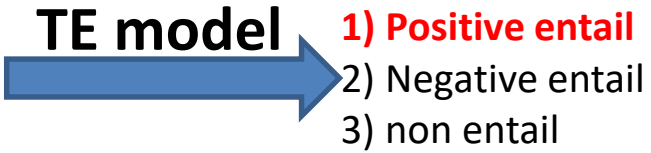
- **Atomic framework**
  - For the automatic discovery of LTE vulnerabilities in cellular networks
  - Through semantic documentation analysis with NLP

  - Input: NAS (Non-Access Stratum) Spec document
  - Output: Vulnerabilities and PoC exploits

Background

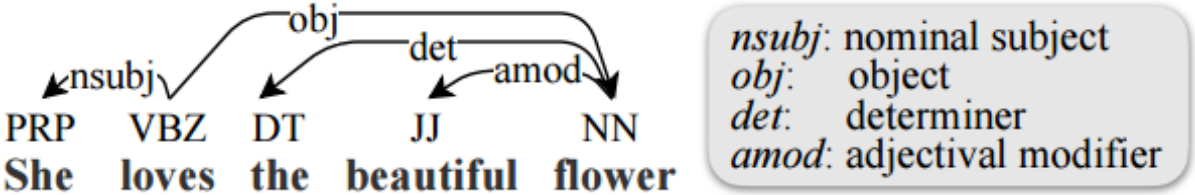
- **NLP (Natural Language Processing)**
  - 1) Textual Entailment
    - A directional relation between a pair of sentences
    - input: a pair of sentences
    - output: positive, negative, or non text entail

sentence1: If you help the needy, God will reward you.
sentence2: Giving money to a poor man has good consequences.



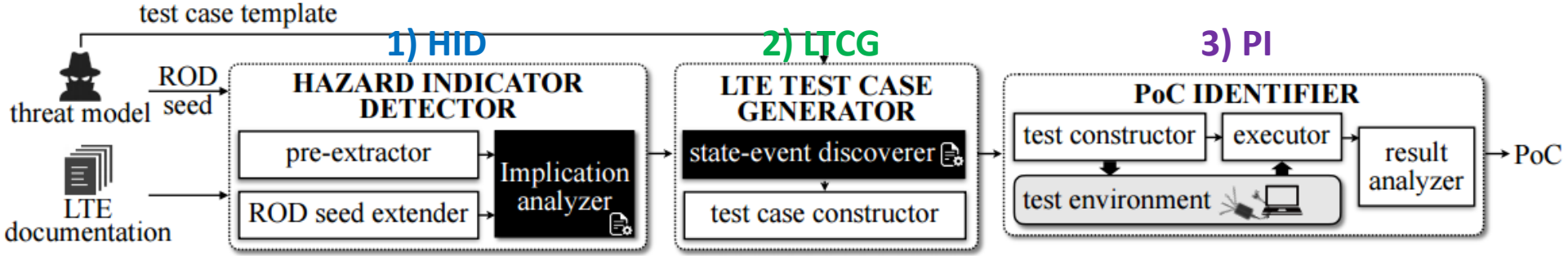
Background

- **NLP (Natural Language Processing)**
  - 2) Dependency Parsing
    - Analyzes the syntactic structure of a sentence
    - input: a sentence
    - output: the grammatical structure and relation



Atomic

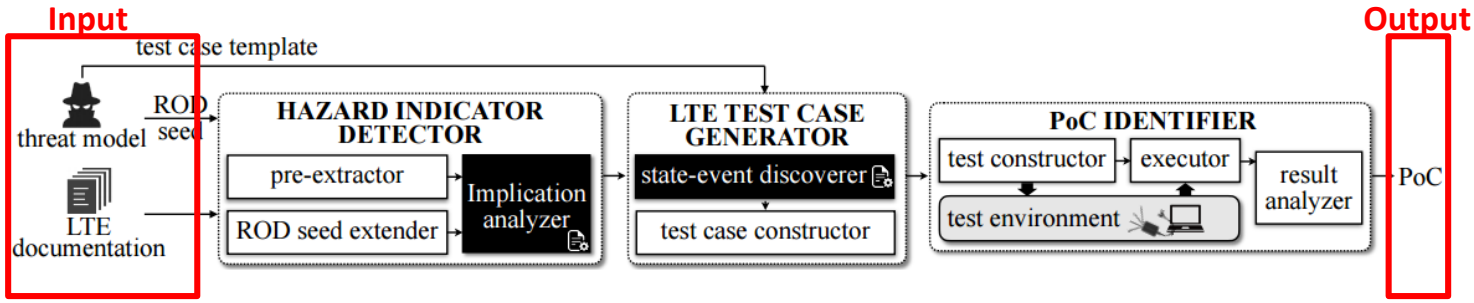
❖ Atomic Architecture



- 1) HID
  - To extend ROD from ROD seed using NLP
- 2) LTCG
  - To recover the state and event from the HI's conditional clause
- 3) PI
  - To determine whether its message triggers a risky operation without proper protection

Atomic

❖ Atomic Input & Output



- Input
  - Document
  - Risky operation description (ROD) seed
  - Threat model
- output: Proof-of-Concept (PoC) exploits

Atomic - 1) HID

- HID (Hazard Indicator Detector)
  - ROD seed: ‘abort procedure’
  - Search for all conditional sentences (if, upon, when, ..)

> ***If*** the network receives a DETACH REQUEST message before the ongoing identification procedure has been completed, the network shall *abort the identification procedure* and shall process the detach procedure

Atomic - 1) HID

❖ HID (Hazard Indicator Detector)

> ***If*** the network receives a DETACH REQUEST message before the ongoing identification procedure has been completed, the network shall *abort the identification procedure* and shall process the detach procedure

– Extend ROD seed

- Find a new verb phrase related to a known ROD through PMI (Pointwise Mutual Information)

$$PMI(x, y) = log \frac{p(x, y)}{p(x)p(y)}$$

PMI('abort procedure', 'process procedure') = 6.7 > threshold (3.97)

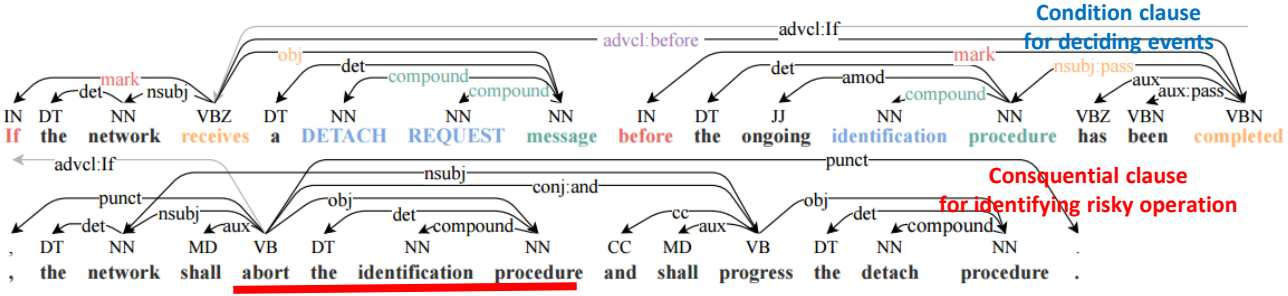
- ex) Find 'process procedure' from 'abort procedure'

Atomic - 1) HID

- HID (Hazard Indicator Detector)

> ***If*** the network receives a DETACH REQUEST message before the ongoing identification procedure has been completed, the network shall *abort the identification procedure* and shall process the detach procedure

  - Extend ROD seed
  - Divide the conditional sentences into conditional & consequence clause using DP



Atomic - 1) HID

- HID (Hazard Indicator Detector)

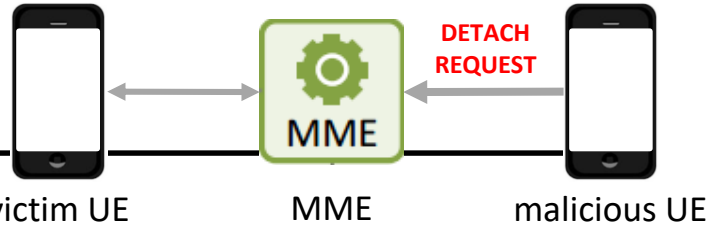
Conditional clause
Premise for deciding events

> ***If*** the network receives a DETACH REQUEST message before the ongoing identification procedure has been completed, the network shall *abort the identification procedure* and shall process the detach procedure

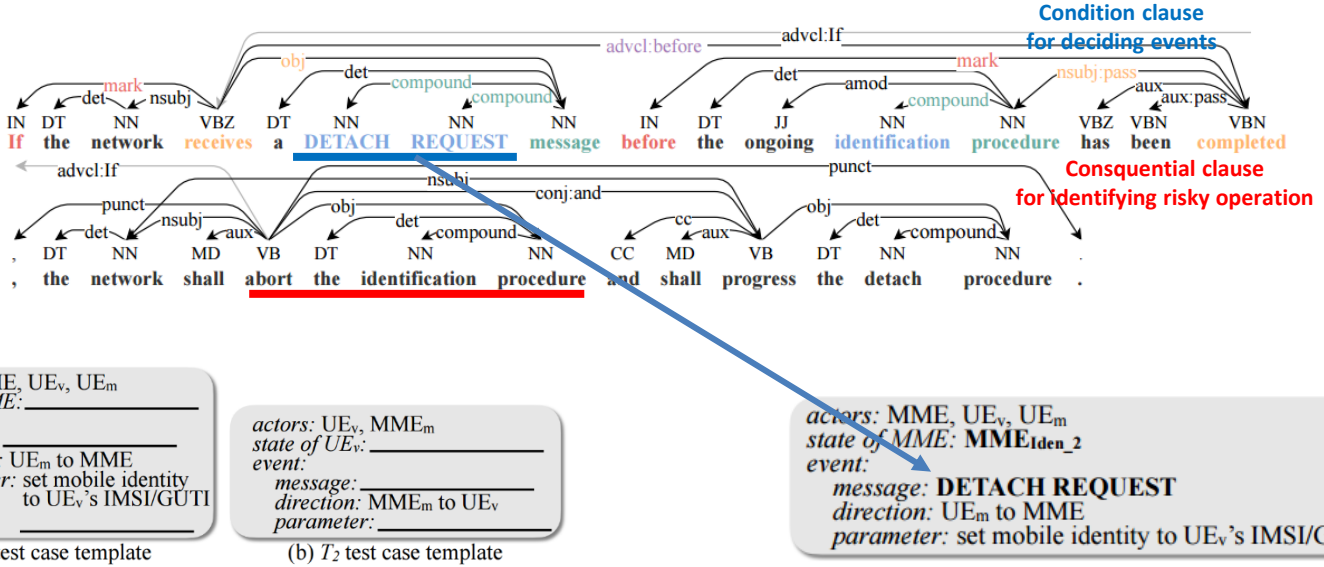
Consquential clause
Premise for identifying risky operations

- Extend ROD seed
- Divide the conditional sentences into conditional & consequence clause using DP
- To identify semantic entails messaging events and risky operations using TE
- From 13,598 sentences, find 5,652 conditional sentences and find 192 HIs

Atomic - 2) LTCG



- LTCG (LTE Test Case Generator)
  - To find out whether the risky operation stated in an HI
  - Fill in the test case blanks from document using DP



Atomic - 3) PI

- PI (PoC Identifier)
  - Executes it to find out whether its message indeed triggers a risky operation
    - 1) Runs an LTE test environment
    - 2) Leverages the hooks implanted in the simulators
    - 3) Keeps track of all communication
    - 4) Inspects the log file
  - If vulnerable, outputs the PoC

Findings

- 65 second per one test overall procedure case

- Atomic finds 42 vulnerabilities in 5 hours
  - 10 in T1, 32 in T2 model
  - 10 in T1 have never been reported before.
  - 15 of them are design weaknesses

Limitation

- The HI should
  - Be included in single sentence
  - Or within a well-formatted multi-sentences
  - Or to be written implicit description, **not explicit**!

- If the risky operation is implicit and not described as a verb phrase??
- Event have been scattered far away from its sentence??
- The effective cross-sentence analysis is possible from its sentence??
- or located in another part of the documentation or a different document??

Future work

- Extend from only LTE NAS protocol
- Extend from only two DoS threat model

Conclusion

- To overcome the ad-hoc manual analysis on the cellular network
  - Find the vulnerabilities automatically with NLP
  - Test LTCG automatically from HI's conditional clause
  - Find out whether its message triggers a risky operation
  - 42 vulnerabilities from 549 page LTE NAS document in 5 hours
- Limitation
  - Only for LTE NAS, for simple two threat models
  - should be included in a single sentence or well-formatted sentences
  - should be written implicit description
- But
  - Extend threat models and documents with advanced NLP??

Before/After work

- Before work
  - Touching the Untouchables: Dynamic Security Analysis of the LTE Control Plane(S&P 2019)
    - A set of security properties are identified from the 3GPP standard to guide the selection and mutation of messages injected into a Long-Term Evolution (LTE) network

- After work
  - DoLTEst: In-depth Downlink Negative Testing Framework for LTE Devices (Usenix 22)
    - Stateful negative testing : tests the content by defining negative testing that is not properly defined in the specification

Questions

- 1. (김한나) I think providing key insights in the technical documentation for the protection and identifying potential security flaws is essential and important to developers. But this paper says that it can be exploited for the attack. Is there any protection method to convey useful information to the developers but cannot be used as vulnerability?
  - This is not a good question, because the specification or the design should not have vulnerabilities. If so, even with NLP, you cannot find any vulnerabilities.
  - However, the question has an interesting twist.
  - In other words, can we design a document that reads well to humans but makes NLP system fail?
  - Yes, it is possible. Please refer to this paper.
    - 'Bad Characters: Imperceptible NLP Attacks', https://arxiv.org/abs/2106.09898

Questions

- 2. (김경태) Even human export with domain knowledge takes several months to understand to specification correctly or well. What is the major limitation or future challenge of this approach?
  - I mentioned the limitations of this approach in page 16.
  - In addition to those limitations, I believe domain expertise cannot be solved using NLP.

Questions

- 3. (오범석) LTE documents are written in natural language, and that's why people try to use NLP. However, NLP is still-developing technology, which means that it cannot perfectly analyze the document to find vulnerabilities. In this sense, can standard be written in computer's language for better implementation and security? For example, there are lots of implementation bugs in several devices like smartphones and I believe that one problem is that it is because standard is written in human's words. Also if computer can automatically find vulnerabilities, I think it will be the best standard.
  - Humans cannot think like a computer. Therefore, there will exist translation error always. Even if we write specification in computer language, we have to translate our idea to a computer language, during which they can make mistake.

Thank You!