Vulnerabilities in O-RAN Software Community project

Research Project Master Computer Science & Engineering

Supervisor: Prof. Dr. Andreas Grebe

Henrik Wittemeier
matr-no. 11157323

Technische Hochschule Köln
Faculty of Information, Media and Electrical Engineering

December 17, 2023

Abstract

5G Mobile Networks are playing a critical role in the modern telecommunication infrastructure. High complexity and short release cycles prevent short manufacturers to enter the market. The introduction of Open RAN aims at an improvement of this situation through cutting the Radio Access Network (RAN) into smaller pieces and create open specifications. The introduction of new technology always has a cost. More interfaces, which are specified by the O-RAN ALLIANCE, lead to an higher attack surface.
The goal of this research is to find vulnerabilities in the Open RAN implementation of the O-RAN Software Community. To reach this goal a lab environment with the main components of the RAN is used. Possible vulnerabilities are taken from studies about the Open RAN specifications and tested or verified if they exist on the lab environment.
The result of the research is that optional security which is specified is not implemented in the source code of the analyzed interfaces and components. Whether traffic encryption nor authentication is implemented on A1 and E2 interface. Outdated software is used for deployment and development processes do not deliver optimal security by using hardcoded credentials and certificates. The O-RAN Software Community should change the focus to higher security standards to ease the use of components for secure development.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
ARP	Address Resolution Protocol
DoS	Denial of Service
JSON	Java Script Object Notation
Near-RT RIC	Near-Realtime Ran Intelligent Controller
Non-RT RIC	Non-Realtime Ran Intelligent Controller
OOM	Operations Manager
RAN	Radio Access Network
RIC	Ran Intelligent Controller
SCTP	Stream Transmission Control Protocol
SMO	Software Management and Orchestration
TLS	transport layer security
UE	User Equipment

Introduction

5G Mobile Networks are playing a critical role in the modern telecommunication infrastructure. High complexity and short release cycles prevent short manufacturers to enter the market. At the moment only the manufacturers Huawei, Ericsson and Nokia are providing large scale mobile network technology [1]. Due to legal restrictions the usage of Huawei technology is very limited in many countries [2]. The result is an oligopoly with a very small amount of competition, which typically leads to a weak price–performance ratio. The introduction of Open RAN aims at an improvement of this situation through cutting the RAN into smaller pieces and create open specifications. It is easier to develope smaller components with specified interfaces instead of an whole mobile network solution. That eases the market entry for smaller companies and improves the market diversification.
The introduction of new technology always has a cost. More interfaces which are specified by the O-RAN ALLIANCE lead to an higher attack surface. In this early state of Open RAN it is necessary to take possible security issues very serious otherwise the elimination would take a lot more effort.
This research is part of the FORAN research project which deals with forensics in Open RAN. FORAN is an cooperation between PROCYDE GmbH and TH-Köln. The goal of this research project is to create an framework that is able to detect malicious activity in Open RAN networks. Part of the research is to identify vulnerabilities in the Open RAN that can be used to generate forensic artifacts that can than be detected and analyzed.
This research project focuses on the identification of vulnerabilities in the Open RAN implementation by O-RAN Software Community. There are many studies published related to Open RAN, but they all concentrate on the specifications. However this concerns are taken to help, to identify possible implemented risks.

OpenRan Architecture

The main difference in the Open RAN infrastructure compared to the older 3GPP standard is to split monolithic mobile networks into parts which interconnect through open source specified interfaces. The Open RAN specifications divide the RAN into smaller components by introducing the Ran Intelligent Controllers (RICs) and splitting the Access Network into control plane and data plane. These components are build to benefit from scalable virtualized platforms with no need for cost intensive custom build hardware.

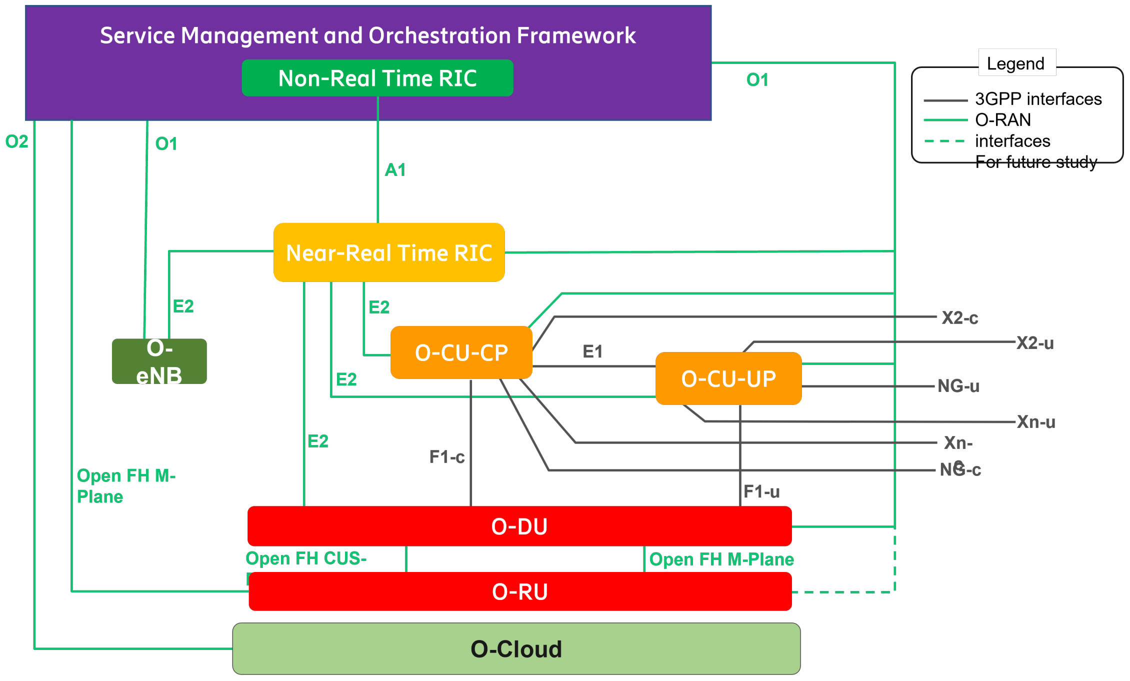

Figure 1: Overview over Open RAN Architecture (Source: O-RAN Software Community)

The research of this project is focused on the interfaces introduced by O-RAN ALLIANCE. The RAN which is used to exploit vulnerabilities is based on the implementation by the O-RAN Software Community. In this project only the RICs are deployed. Missing components which use the E2 interfaces are replaced by an E2 Simulator.
The Open RAN project separates its functions based on their time criticality. Functions that need a time accuracy of higher than 1 second are handled by the Non-Realtime Ran Intelligent Controller (Non-RT RIC). Functions with a timerange between 1 second and 10 milliseconds are handled by the Near-Realtime Ran Intelligent Controller (Near-RT RIC). Applications with higher time requirements (<10ms) are handled by

the O-DU in real-time [3].

2.1 Non-Real Time RIC

The Non-RT RIC automates the management of RAN functions. Its focus are non-realtime management functions such as radio resource management, higher layer procedure optimization and policy optimization [4]. It has functionality to provide trained machine learning models beside parameters, guidance and policies to the Near-RT RIC. All communication to the Near-RT RIC works via the A1 interface.

2.1.1 A1 Policy Management Service

The A1 Policy Management Service maintains a repository of all configured policy instances in the network. A policy instance is a derivation of a policy type which is assigned to a Near-RT RIC or an rApp. RANs can consist of multiple Near-RT RICs and rApps. Each policy type can be applied to a single User Equipment (UE), Near-RT RIC or globally for all objectives. If a Near-RT RIC fail or reboot causes inconsistencies the Near-RT RIC can re-synchronize policy instances from the A1 Policy Management Service [4].

2.1.2 rApps

rApps are a tool to make a Radio Access Network configurable by needs. They split Network control functions into separate independent parts with a specified interface. rApps will include control loops that are based on machine learning algorithms or other control loop regulators. rApps communicate via the R1 Interface with the Non-RT RIC. They mostly work by collecting data from cells or radios and make rule or Artificial Intelligence (AI) based decisions to improve network performance. Example use cases for rApps are energy and performance optimization and diagnosis [5].

2.2 Near-Real Time RIC

The Near-RT RIC gets policy guideline from the Non-RT RIC and returns policy feedback. It controls RAN elements and resources with optimization procedures that take 10 milliseconds to 1 second [6].

2.2.1 xApps

While the RIC is a framework which provides interfaces for the controlling of the RAN it has no control functions implemented. To give the Near-RT RIC a functionality xApps are used. An example for a set of xApps are traffic steering with quality of experience prediction and anomaly detection.

2.2.2 A1 Mediator

The A1 Mediator receives Policy guidance from the Non-RT RIC. Every change done to a policy will be distributed to the A1 Mediator. After receiving, policies are forwarded to the Near-RT RIC and the xApps [7].

2.3 Interfaces

All interfaces are described by open standards to maximize compatibility and facilitate a multi vendor RAN

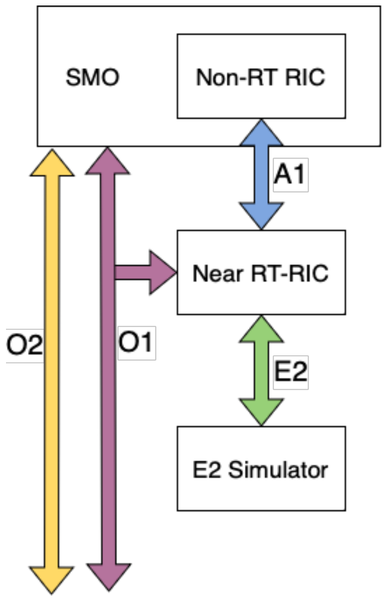

Figure 2: Open RAN Interfaces in lab setup

2.3.1 E2

The E2 interface interconnects the Near-RT RIC with the O-DU, the O-CU and the O-eNB. It is used to control RANs by transmitting policies and guidelines. The controlling can be performed on different levels: For each UE, for a group of UE, cell by cell and for an entire area of cells.
Beside RAN performance optimizations the E2 interface is used to configure and retrieve measurements. To retrieve data UE can be requested to report L1/L2/L3 measurements [8]. The protocol stack of the E2 interface is based on the E2 Application Protocol (E2AP) over SCTP. The protocol stack has no transport layer security (TLS) implemented. [9].

2.3.2 A1

The A1 interface is used to enable the provision of policies from the Non-RT RIC to provide guidance, ML model management and enrichment information to the Near-RT RIC. A simple feedback mechanism provides information about the state of policies. Feedback for the impact of changes on the A1 interface is received over the O1 interface. Policies published by the Non-RT RIC applicate usually to UEs and cells [10].
The interface is based on a RESTful API, which exchanges Java Script Object Notation (JSON) formatted policies. The protocol stack is based on TCP/IP, HTTP with TLS and authentication through OAuth 2.0. Encryption and Authentication are included in the specification. To enable a bidirectional connection Non-RT RIC and Near-RT RIC can both act as client and server [11].

2.3.3 O1

The O1 interface interconnects all O-RAN managed elements within the Software Management and Orchestration (SMO) Framework. It uses the protocols SSH, HTTP and NETCONF. It delivers management services for provisioning, fault supervision, performance assurance, tracing, file management communication surveillance and more [12].

2.3.4 O2

The O2 interface establishes a secure connection between the SMO part of the Non-RT RIC and the O-Cloud. It is specified variable so that new functions can be implemented without changing the specifications [13].

2.4 Lab Setup

The evironment build for this reserach is based on the O-RAN Software Community implementation in H-Release. The Non-RT RIC an Near-RT RIC are deployed in two seperate kubernetes singlenode clusters (see Figure 3).

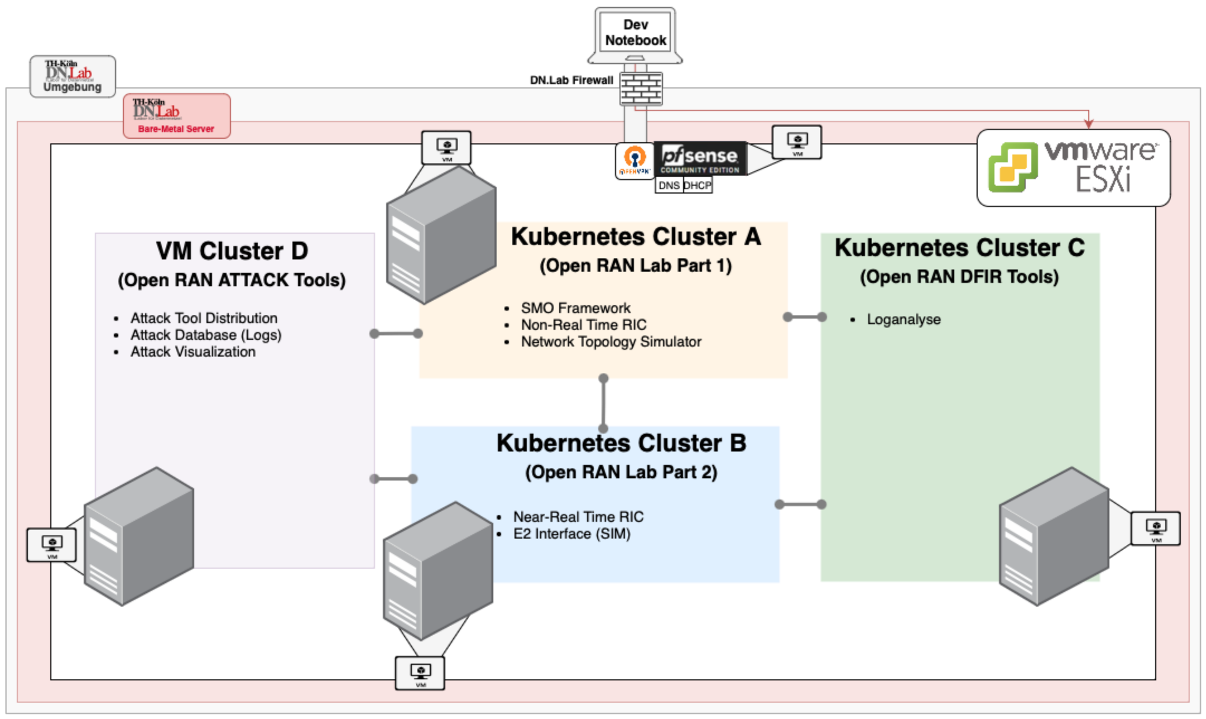

Figure 3: Lab environment structure (Source: FORAN TH-Köln)

All machines are running inside an VM on the same ESXI hypervisor and have network connectivity with 1 Gbit/s. All attacks are launched from the VM Cluster D. The entire setup is placed in an own network behind a firewall with acces to the internet. Access to the network from outside is gained with an VPN connection. The Kubernetes cluster C hosts some log analyzing tools which are not used in this project.

Methodology and Scope

This research project has its scope on the Open RAN implementation from the O-RAN Software Community. The considered release is the H-Release. The tests are performed against the implemented software, not against the specifications. Specifications are only consulted to give recommendations for future improvements or find attack surfaces.

3.1 Attacker roles

To estimate the criticality of given attacks it is necessary to examine what attacker has the potentiality to perform an attack. The different attacker roles can be considered hierarchically.

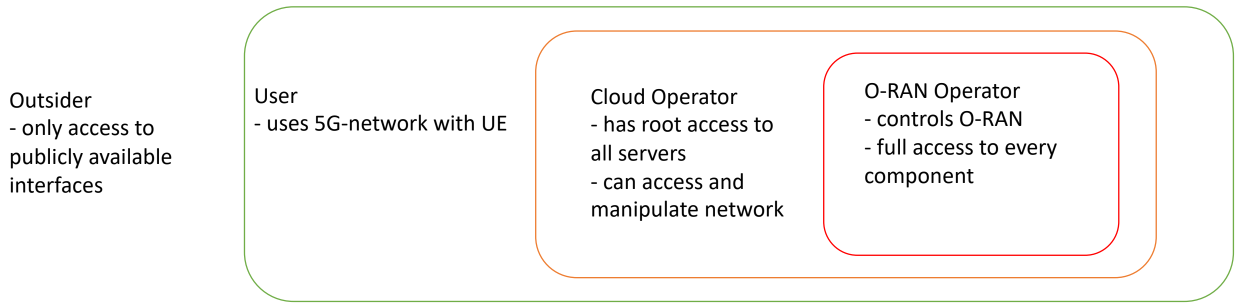
Outsider
- only access to publicly available interfaces
User
- uses 5G-network with UE
Cloud Operator
- has root access to all servers
- can access and manipulate network
O-RAN Operator
- controls O-RAN
- full access to every component

Figure 4: Attacker roles and their accessibility

The O-Ran Operator is the most powerful attacker role, as visualized in Figure 4 the role has full access of all O-RAN components and inherits aspects from all other attacker roles. All attacker roles should not be taken in the literal sense. The role ”Cloud Operator” could also be an employee that has unlimited access to the same resources as the ”Cloud Operator”. Also attacker can gain privileges and exceed their limitations, in this study however every attack is assumed to be the first move without prior access that would exceed their described boundaries.

3.2 Protection goals

Everywhere a successful attack can be executed, a protection target violation is reached. To categorize attacks following protection goals are defined:

3.2.1 Confidentiality

The goal confidentiality describes that everything that is stored or transmitted inside the Open RAN is kept secret. User data or data which can be used to draw conclusions about users should be handled with extra care.

3.2.2 Integrity

Integrity is the goal of counterfeit protection. To identify if a message is counterfeited, encryption can be used. If a message has changed decryption should fail. Integrity is demanded for all kinds of traffic. Otherwise Users could be feinted or functions of the RAN could be manipulated.

3.2.3 Availability

Mobile Networks are critical Infrastructures. In times of machines and humans with strong dependencies to mobile Networks it is urgent to secure the availability of mobile networks. Availability is not a binary state, attacks which decrease the performance have also an impact on the availability.

Related Work

In the last years many studies about vulnerabilities and weaknesses in the Open-RAN specifications where published. Even if this research handles the implementation by the O-Ran Software Community, an overview over the research about the Specifications gives a good overview over the possible evaluation targets. The aspects reviewed in this studies are scoped to the components stated in Figure 1.

4.1 Open RAN Risk Analysis - BSI

The Open RAN Risk Analysis research gives a very detailed survey over vulnerabilities in the O-RAN Specifications. All vulnerabilities are evaluated for the protection goals confidentiality, integrity, accountability, availability and privacy. All results are compared to the 3GPP specifications to show improvements or degradation of cybersecurity in the mobile networks [14]. General recommendations are that security protocols that are used should have least vulnerabilities as possible. To ensure integrity and privacy of all data it is necessary that data is protected at transport as well as at rest so that attackers or insiders with filesystem access are limited in violating privacy regulations [14]. More specific to interfaces in the Open RAN infrastructure it is necessary that a role concept should be implemented [14]. Open RAN has many powerful components as the O1 interface which has connection to many interfaces, all components should only be granted the least privileges needed for functionality, to decrease the impact of malicious components. Also a role concept for xApps and rApps is necessary to ensure that malicious Apps cannot affect all components that are connected to the respective interface [14].
With the multi vendor approach of Open RAN it is necessary to separate each component as much as possible to protect other components from beeing compromised. Also no manufacturer or actor in the RAN should be considered as trusted whether RAN operators nor cloud or network operators [14].
While many types of attacks can be prevented, DoS attacks can not be entirely prevented so its important that DoS attacks have limited impact on the performance of the entire RAN [14].
The A1 interface optional can be secured through a TLS session. Thereby that the A1 interface mostly transmits policies and RAN steering data the protection goals confidentiality and integrity are not violated but the network performance for users could be decreased significantly [14].
Unencrypted transmission on the E2 interface lead to high risk to all stakeholders, in the specification however encryption is only an optional requirement that the e2 interface is secured by tunneling the connection through IPSec. The E2 interface has a relevant role in Near-RT communication of the RAN so a DoS attack have a high influence on the performance of the whole system [14].
The O1 interface uses the NETCONF protocol over SSH. SSH is a very powerful protocol that could give

system wide access in regard of misconfiguration. In addition to that the specification requires a backwards compatibility to prior vulnerable versions of SSH [14].
With an attack on the O2 interface it may be possible to control the entire RAN. Weaknesses in the specification cannot be addressed because the specifications are not finalized at this point [14].
xApps which could be delivered by different vendors are able to access the E2 and A1 interfaces. rApps can Access the O1 and O2 interface. The concept of the apps is that every app has a dedicated functionality, but have no restricted Access to the interfaces.

4.2 Security considerations of Open RAN - Ericcson

The environment of an open specified RAN introduces many chances for improvements. One big opportunity is the market diversity that will increase with effects as higher price-performance ratio. Implementing a RAN by picking the best vendor for each component introduces new challenges [15]. Changes to Specifications and according updates must be tested across all vendors with all component versions. Vendors that want to deliver interoperable components have to communicate with other vendors [15]. Due to the introduction of AI and Machine learning new vulnerabilities could be introduced. Malicious input data could be indistinguishable to random noise and models could be manipulated [15]. The development of open source software also has chances and risks. Many developers can analyze the source code and find vulnerabilities. A risk is that a malicious developer contributes malicious code or backdoors that are deployed to production systems [15].

4.3 Open RAN Security Report - Quad Critical and Emerging Technology Working Group

Due to its division in smaller components Open RAN has more interfaces and therefor a much higher attack surface than the 3GPP standard. Every component of the Open RAN specification has related security threats. Most of the vulnerabilities come through specification gaps or inconsistencies in the hierarchical relation of the specifications. The Open RAN Security Report has a strong focus on AI/ML related risks as described in section 4.2.
Security Issues pointed out:

- Malicious xApps can exploit UE identification, track UE location an change priority (DOS).
- Missing privilege restriction in Near-RT RIC.
- A1 Untrusted peering, sniffing, modification.

The report ends with a security checklist with concrete vulnerabilities [16].

4.4 Summary of vulnerabilities

From the studies above the list of vulnerabilities which may be exploitable in our virtual RAN environment is created. As many researches overlap in their realisations they are not considered further

Component	Description	Protection Goal	Attacker	Impact
A1	Read and write without authentication to A1 interface	Confidentiality	Cloud Operator	Medium
A1	Denial of Service on A1 interface	Availability	Cloud Operator	Low
E2	Read and Write messages on E2 interface	Confidentiality	Cloud Operator	High
Inter-Component	Influencing the performance of a service from another service	Availability	RAN Operator	High
Repository	Malicious Code Injection	Integrity, Availability, Confidentiality	Outsider	High

Table 1: Choosen Vulnerabilities to be respected in following research

These vulnerabilities are can be addressed in our research environment. Other vulnerabilities e.g. O1 and O2 interface vulnerabilities are not listed because those interfaces are not fully present in our environment. Vulnerabilities related to xApps and rApps where also not considered because no working apps where found for the H-Release.

Vulnerability exploitation

In this chapter the vulnerabilites specified in Table 1. Are verified and attacked if possible. Vulnerabilities that cannot be tested are confirmed by analyzing the source code repository or by analyzing traffic and behaviour in the lab setup.

5.1 A1 read and write

The A1-interface is an http RESTful API with endpoints on the Non-RT RIC and Near-RT RIC. Due to the fact that the interface does not use TLS a simple API call can be done to read a policy from the A1 interface. The A1 interface on the Near-RT RIC must be accessible from the Non-RT RIC and vice versa to enable communication. This means that an attacker with access to the networking between the RICs could execute this attacks. Attackers that have this access are at least cloud providers. But this attack could also be a part of another attack which already gained access to the networking of the datacenter or machine.

5.1.1 Policies

The policies delivered to the RICs are stored in an policy database to be accessed by xApps and rAppps. The policies can be used to configure the behaviour of xApps during runtime. The example in appendix 8.1 is used to configure a traffic-steering xApp.

5.1.2 Read

To receive a policy a GET request to the url `http://<IP>:30091/a1-policy/v2/policy-types/<policy-id` where the parameter IP is replaced with the IP address of the Non-RT RIC. The parameter policy-id must be replaced with an integer of a policy that was previously created. The return of that request is an JSON object which represents the policy.

5.1.3 Write

To add a policy to the A1 policymanager a POST request to the url `http://<IP>:30091/a1-policy/v2/policy-types/<policy-id` with a json payload including the policy must be performed.

5.1.4 Delete

To delete a policy a DELETE request must be executed on the url `http://<IP>:30091/a1-p/policytypes/<policy-id>`.

5.1.5 Impact

The impact of policy insertions, modifications, deletions is very dependent on the usage of that policy in an xApp. In this case a traffic steering xApp could be manipulated that traffic is steered in a way that it causes a severe performance degradation of UEs. The performance of a mobile network and its cells could be degraded for example by triggering a lot of handovers that would cause an high management overhead.

5.2 A1 DoS

In contrary to the previous attack which could potentially degrade the whole Mobile Network the following attack focuses on the A1-policymanagement component, which possibly have further influence to the mobile network but is not primarily focussed on. The main focus is to make the A1 interface unreachable.
This test performs an Denial of Service (DoS) attack. The goal of an DoS is to make a service become unavailable or crash. DoS attacks try to utilize all of a certain resource so that related processes crash due to timeouts. DoS can be reached through penetrating lower network protocol layers like TCP or use a service in its intentional way at a very high frequency which is the case in this approach.
To test which impacts a DoS can have on the A1 interface, tests with two different strategies are performed.

5.2.1 POST small but many policies

First attempt is to POST many small policies to the RIC so that the CPU load of the server should increase significantly. The result of the test is mostly dependent on the computing overlap that is utilized by the API to store the policies. The execution is similiar to the execution described in subsection 5.1.3. To achieve as many requests as possible in parallel 16 threads are creating policies simultaneously. To prevent that policies are dropped it is necessary that each policy has its unique policy-id.

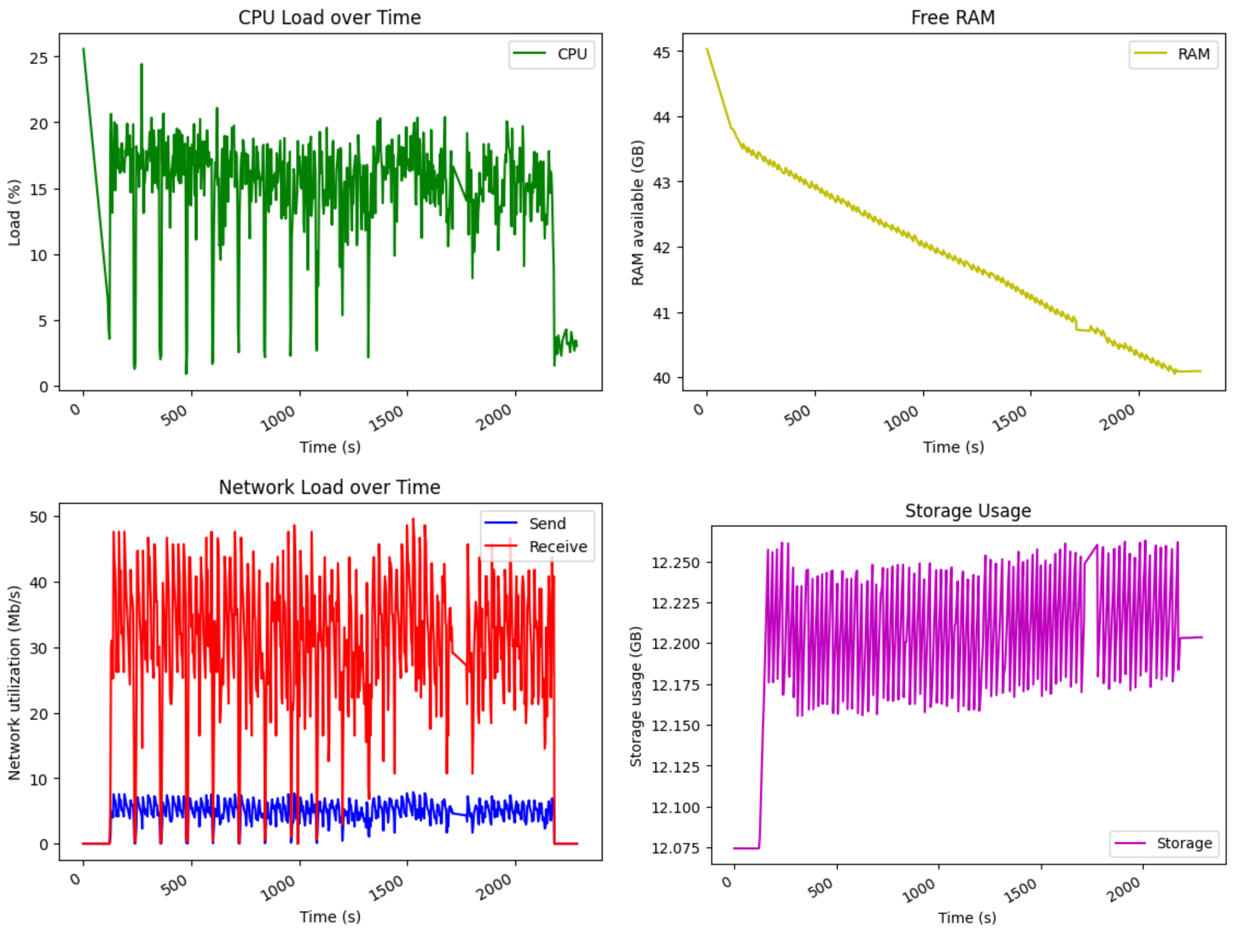

Figure 5: Small Policy DoS

The graphs in Figure 5 show the resource allocation that occurs during the attack. It is visible that the amount of free RAM decreases over time at a very low ratio. All other resources do not show significant changes or particularities. The CPU load is at 15 to 20 % which should not be a concern. The network is used at a receiving rate around 35 Mb/s. This test does not cause a performance degradation so is not a successful DoS.

5.2.2 POST large but few policies

The previous test did not cause a DoS but had the result that the RAM usage increases with more policies that are saved to the database. The policies send to the A1 interface are not persisted in storage but are placed in a Redis database. Redis databases store all their data in memory to ensure high speed accesses.
In this approach we build a very large policy that has around 1GB of text. This is possible because the ”description” field of the policy is not limited in its size.

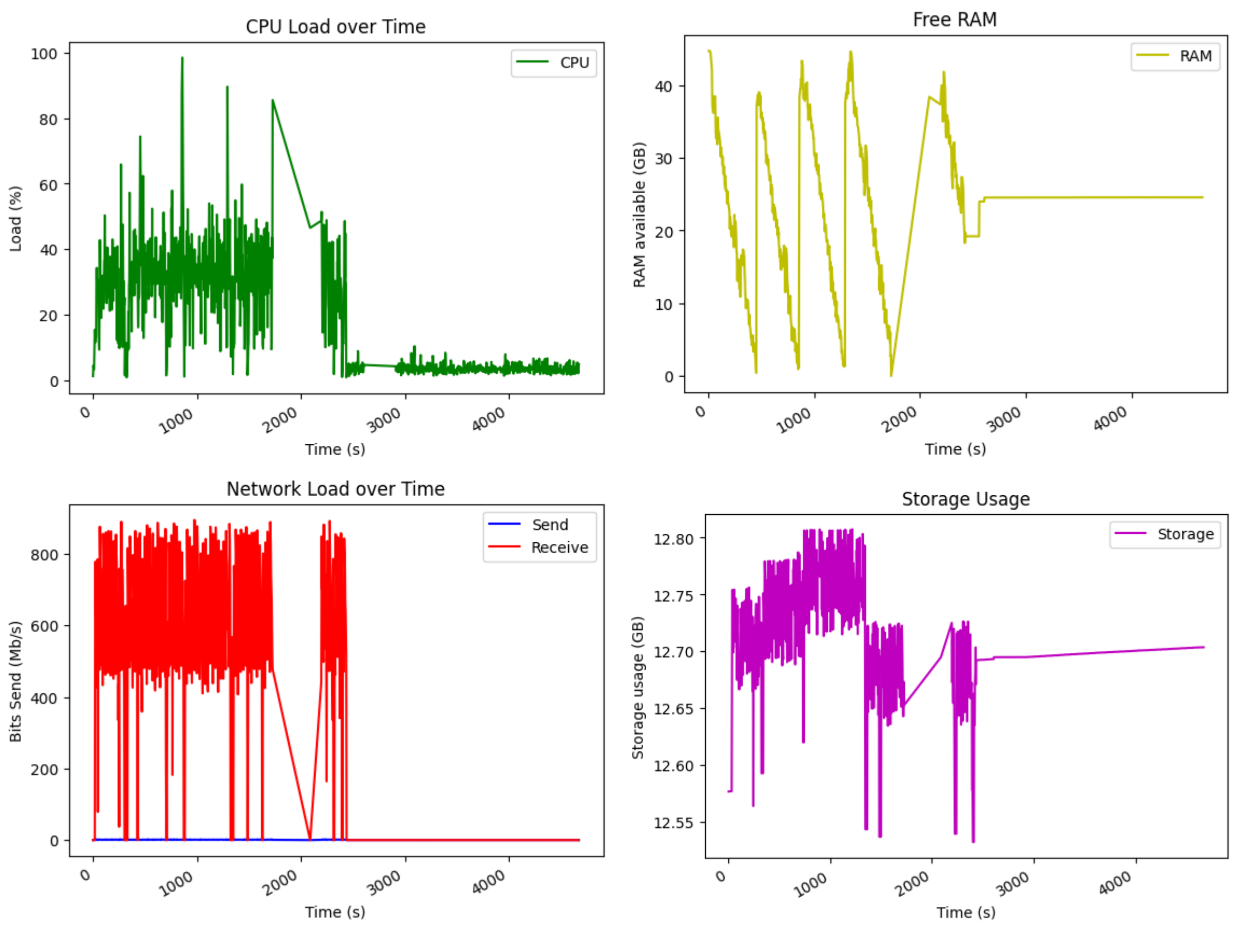

Figure 6: Large Policy DoS

Sending the large policy causes a very different resource allocation. The CPU load and the storage usage do not differ significantly to the test above. The network throughput reaches around 650 Mb/s. The high wobbling of the network graph could come from network congestion. Much more of interest is the RAM graph. The sawtooth like function indicates a problem that is produced. The big config is written directly to ram. The data that is written to RAM is endless while the RAM is not. When the Redis database is overflowing because no more ram can be allocated it is completely dropped. Another point in the diagram in Figure 6 is that the peaks are getting higher after the Database is dropped, that shows that other components are also enforced to lower their RAM usage.

5.3 E2 Interface

The E2 interface is an Stream Transmission Control Protocol (SCTP) based connection that enables communication between the Near-RT RIC and components of the RAN. The O-RAN Software Community uses the implementation as defined in the specification [9]

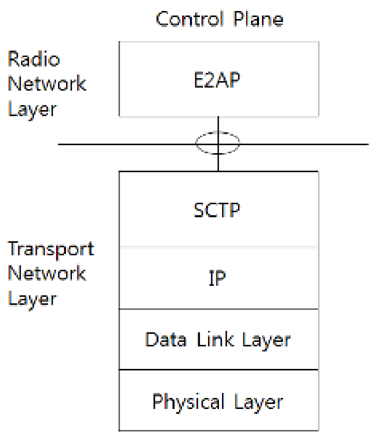

Figure 7: E2 interface protocol stack

No layer in the protocol stack of the E2 interface implements encryption thereby the plan for future specification releases is to encapsulate the E2 interface in IPsec. In the H-Release of the Software Community it is not implemented. The E2 interface is accessible and also no authentication is required to establish communication. The E2 interface is very powerful and has a central role in the Open RAN. Attackers which must be at least Cloud Operators have the ability to violate all protection goals. Since SCTP is not as common as TCP and UDP it was analyzed for vulnerabilities. The SCTP processing is done in the kernel which means that vulnerabilities would not come with bad applications but with outdated operating systems. As we use a recent operating system the kernel is kept up to date automatically so no vulnerabilities can be found.

5.4 Inter-Component interference

The implementation deployed in the research environment separates all components to Containers. Containerization's goal is to separate each function into an own container to make access restriction to resources or other processes possible. The main goal is that after an attacker has reached access to one component it is not possible to resume the attack on the host system. To examine if an malicious or infected container can influence other components in their behaviour attacks from inside the container are performed. To execute this attack as a point of start the privileges owned must be at least Ran-operator.

5.4.1 Arp-Spoofing

Arp-Spoofing is a very simple but powerful attack that happens on Network layer-2. The Address Resolution Protocol (ARP) is used to learn to which mac-address a packet should be delivered based on its destination IP-Address. To establish a connection the sender asks "who has `<IP-address>`" and a participant which uses that IP-Address or has it in its routing table answers that request with its mac-address.
ARP spoofing simply means that a malicious member is answering every single ARP request so that all other members send their packages to the malicious host. It would be possible to imitate the gateway so that traffic that would usually go to an update server is routed through the malicious network member.
If the malicious host simply ignores the traffic it would lead to a DoS because all packets in the network would be dropped. If the malicious host routes the packets to their originally destination the change would not be observed by other components, but the malicious host is able to sniff that traffic and also change it if

its not encrypted.
In summary the three protection goals availability integrity and confidentiality would be violated.
For this test the open source command line tool ”arpspoof” is used. [17].
By sending the mac-address of our own component to all ARP requests of specific hosts we are able to redirect traffic to our component so the network abstraction of the kubernetes cluster does not prevent arp-spoofing and the attack where successfully.

5.5 General Project Issues

In this section issues are addressed which represent general issues in the development process of the open source project.

5.5.1 Hardcoded certificates

The repository uses hard coded certificates for the SMO part from ONAP. These hard coded certificates can be used by an attacker to fake its identity. Connections which are TLS encrypted normally grant integrity and confidentiality. By using the same certificate as the user it is possibly to decrypt those connections change context and encrypt them again. This violates the protection goals integrity and confidentiality. If interfaces are routed through the internet because they are TLS encrypted the attacker role outsider could execute this attack.

5.5.2 Hardcoded Credentials

The Operations Manager (OOM) is a component of the Open RAN deployment which uses hard coded credentials. In the OOM wiki a list with around 50 hardcoded credentials is published. These credentials can be used by an attacker to gain access to password secured components. Attacker roles that can use this attack could even be outsiders depending on the network access to the components. The impact of this attack is depending on the components that could be accessed [18].

5.5.3 Branch protection rights

The main repository for the deployment of the Non-RT RIC is the repository ”it/dep”. Changes to the main branch can not be done directly but through merging a pull request. A part of the git merge process is a review that can be configured. The review is necessary to ensure that not everyone can contribute its code to the repository. Reviews only make sense when they cannot be performed by everyone. The Gerrit instance which hosts the O-RAN repositories also implements review rules. Most merge requests to the repository have at least one reviewer. But there are also commits that are only reviewed by the pusher himself.
This means that if a single developer becomes an attacker or an attacker gains access to repository credentials it is possible that the developer contributes malicious code without any restrictions. In times of CI/CD it is possible that those changes are deployed to an production system if no other safety precautions are taken. This attack could cause all protection goals to be violated because all components can be manipulated.

Results

Mobile Networks belong to the critical infrastructure that should be protected with the highest effort possible. The research shows that this effort should be increased by the O-RAN Software Community.

> ”The aim is to achieve a solution that can be utilized to unify and accelerate the evolution and deployment in the RAN.” [19]

The intended use of this implementation is to support the development of a Open RAN implementation. It is questionable if this implementation helps to build a RAN in terms of security.

6.1 Technical Vulnerabilities

All considered interfaces have only implemented the minimal security requirements that are specified by the O-RAN ALLIANCE which means that there is no security at all. An useful reference implementation should support all maximum security features to ease the development of a secure real world RAN implementation. During the development process of different components it would have an increased use if new implemented components could be tested in the existing reference implementation. This can only be achieved if the new components do not support the optional security standards. The result is that this reference implementation only helps developing insecure components.
Another point that should be improved is the stability of the system. Before a database is dropped, other components are forced to decrease memory usage or the application crashes there should be mechanisms which prevent that it comes to critical resource allocations.
The fact that the developers choose to implement the RIC in kubernetes is contemporary to achieve a scalable RAN. As above the implementation in kubernetes also uses the least restrictions that are possible which causes a lack of security. To limit the impact of malicious components a way to go would be to not grant full permissions on network interfaces but instead restrict them to the needed features. The Near-RT RIC also uses an 4 year old Kubernetes version (v1.16) which had its end of life 3 years ago. This adds other vulnerabilities that could be avoided easily.
Even if the real world usage is not a criterion that was taken into account during the development, developers that try to create a real world Open RAN will take the reference implementation as an assistance. To ensure that they concern on security right from the start the O-RAN Software Community implementation should be changed from least necessary security to the most possible security.

6.2 Structural Vulnerabilities

The low effort approach which was described in section 6.1 is also applicable to other security areas. The developers choose to use hard coded credentials for all components. This may ease the test access but introduces big vulnerabilities. Its not new that developers use hard coded credentials in systems which are in a testing stage but forget to replace those credentials when they are delivered to production use. Even if the O-RAN Software Community project is intended for test uses only it is possible that developers take components out of that project to publish them in their environment. Hard-coded credentials should be avoided from the start of a project to eliminate this problem.

The same applies to hard coded certificates. It would be very easy to build the certificate chain at install time to ensure that each development setup has its unique certificate and it cannot be used by attackers to change their identity. An attacker that has network access and the TLS certificate could perform an man-in-the-middle attack by using the certificate while changing the traffic routing by using arp-spoofing.

The base64 encoded certificates that are pushed to the repository are also not reviewable. It is possible that attackers push a malicious certificate chain to a repository so that they are trusted by the component using that certificate chain for example for updates. This could also be a problem when deploying the code to an test setup. Malicious certificate chains can lead to further insecurities.

Development principles and security of research environments can be improved by requiring at least one review per commit. Commits without reviews can lead to malicious code integrations to the repository. The user of this code would mostly be large telecommunication companies which represent a attractive target for attacks.

6.3 Research

Researching vulnerabilities in systems never lead to an all encompassing result that can rule out the presence of any other vulnerabilities.

This research was performed with a main part of the Open RAN implementation but leaves out any components that take profit from the availability of the RIC. Also no UEs is connected which would add sensible data to the network. Under this circumstances the effects and impact of an vulnerability can only be estimated. It is also not a goal of this research to give an all-encompassing overview over vulnerabilities in the Open RAN implementation. This research's goal is to identify possible vulnerabilities.

The security standards of the components considered in this research a production readiness is ruled out. The specifications of the O-RAN ALLIANCE show many inconsistencies in the definition of security standards that should be applied. The O-RAN Software Community however chooeses the lowest effort approach when it comes to security which introduces many needless vulnerabilities. The implementation should not be used as an ideal to reproduce in security related terms.

Conclusion

To examine vulnerabilities which are present in our development setup, related work is analyzed for vulnerabilities. As there is no related work that describes vulnerabilities in the O-RAN Software Community implementation we try to transfer the gained knowledge from related work on the O-RAN ALLIANCE specifications. If possible and reasonable, tests against the vulnerabilities were executed in our development environment. The results in that evaluation show that both interfaces have no security algorithms implemented at all. Whether traffic encryption nor authentication is implemented on both interfaces. The software version of Kubernetes which is part of the infrastructure is outdated and components are not secured against mutual disturbance. The O-RAN Software Community developes the reference implementation not according to the maximum specified security standards. Thereby vulnerabilities are introduced to the RAN. It is assumed that all stakeholders that deliver infrastructure and Open RAN components are trusted which is the case in an local test environment. If this implementation or components of it are taken to production the vulnerabilities can be attacked from malicious RAN Operators or Cloud Operators. If a company developes single secure components they cannot be tested in this implementation since no security features are supported. Open specifications and the open source code aim to enable developers to implement single components, to ease the market entry for small companies. Contradictory to that is, that small companies cannot only develope a single secure component and test it in the reference implementation delivered by O-RAN Software Community.

Through this conflict the goal of a reference implementation is missed.

Also there are weaknesses in the O-RAN Software Community development process. Hardcoded credentials and certificates can be found in the repo. These lead to unnecessary possible backdoors. As the repository is quite popular the merge and review roles should be overthougt. Developers who can merge unreviewed code to the master branch can implement bugs without notice which lead to unavailability.

7.1 Recommendations

The O-RAN Software Community should focus more of their effort on security related issues. The weak security that is specified by O-RAN ALLIANCE should not be lowered in the reference implementation.

Issues in the infrastructure can be corrected easily by changing access rights for Pods and using latest software versions.

Hardcoded credentials and certificates should be avoided. Generating secure credentials at deployment time would increase the security significantly. Since the implementation has a large number of users, development processes should be improved to lower the risk of malicious code injection.

7.2 Future Work

As this research was initiated in the FORAN research project further investigations should take place to determine if the attacking of some of the vulnerabilities leaves forensic artifacts which can be used to detect malicious behaviour. As this research is based on an simple lab environment and not a fully functional Open RAN 5G Network many effects can only be estimated. To determine actual effects the testing of the mentioned vulnerabilities in an running network would be neccessary. In an complete Open RAN 5G Network also more interfaces and components could be observed and analyzed.

Bibliography

[1] Fraunhofer-Gesellschaft e.V. 5G – NETZE UND SICHERHEIT, 2020.

[2] Reuters. European countries who put curbs on Huawei 5G equipment, 2023. Last accessed 29 September 2023.

[3] Zahid Ghadialy. An Overview of O-RAN Architecture, 2021. Last accessed 29 September 2023.

[4] O-RAN Software Community. Non-RT RIC - Summary, 2023. Last accessed 29 September 2023.

[5] Ericsson. Intelligent Automation rApps, 2023. Last accessed 29 September 2023.

[6] O-RAN Software Community. Scope of the near-RT RIC platform and its components (summary), 2023. Last accessed 29 September 2023.

[7] O-RAN Software Community. Scope of the near-RT RIC platform and its components (summary), 2023. Last accessed 29 September 2023.

[8] O-RAN ALLICANCE e.V. O-RAN Use Cases and Requirements 4.0, 2023.

[9] O-RAN ALLICANCE e.V. O-RAN E2 General Aspects and Principles (E2GAP) 4.01, 2023.

[10] O-RAN ALLICANCE e.V. O-RAN A1 interface: General Aspects and Principles 3.01, 2023.

[11] O-RAN ALLICANCE e.V. O-RAN A1 interface: Transport Protocol 3.0, 2023.

[12] O-RAN ALLICANCE e.V. O-RAN Architecture Description 10.0, 2023.

[13] O-RAN ALLICANCE e.V. O-RAN O2 General Aspects and Principles 1.02, 2023.

[14] Stefan Köpsell, Andrey Ruzhanskiy, Andreas Hecker, Dirk Stachorra, Norman Franchi. Open RAN Risk Analysis. Technical report, BSI, 2022.

[15] Scott Poretsky Jason S. Boswell. Security considerations of Open RAN, 2020.

[16] Scott Poretsky Jason S. Boswell. Open RAN Security Report, 2020.

[17] github.com. arpspoof.py - An ARP poisoning tool, 2014.

[18] ONAP. OOM Hardcoded Passwords List, 2019.

[19] O-RAN ALLIANCE. Open Software for the RAN, 2023.

Appendix

8.1 Example Policy

```
"description": "",
  "policy_type_id": 0,  "name":"1",
  "create_schema":{
      "$schema":"http://json-schema.org/draft-07/schema#",
      "type":"object",
      "additionalProperties":False,
      "required":["class"],
      "properties":{
          "class":{
              "type":"integer",
              "minimum":1,
              "maximum":256,
              "description":"1"
          },
          "enforce":{
              "type":"boolean",
              "description":"1"
          },
          "window_length":{
              "type":"integer",
              "minimum":15,
              "maximum":300,
              "description":"1"
          },
          "trigger_threshold":{
              "type":"integer",
              "minimum":1
          },
          "blocking_rate":{
              "type":"number",
              "minimum":0,
              "maximum":100
          }

      }
  },

  "downstream_schema":{
      "type":"object",
      "additionalProperties":False,
      "required":["policy_type_id", "policy_instance_id", "operation"],
      "properties":{
          "policy_type_id":{
              "type":"integer",
              "enum":[21000]
          },
          "policy_instance_id":{
              "type":"string"
          },
          "operation":{
              "type":"string",
              "enum":["CREATE", "UPDATE", "DELETE"]
          },
          "payload":{
              "$schema":"http://json-schema.org/draft-07/schema#",
              "type":"object",
              "additionalProperties":False,
              "required":["class"],
              "properties":{
                  "class":{
                      "type":"integer",
                      "minimum":1,
                      "maximum":256,
                      "description":"1"
                  },
                  "enforce":{
                      "type":"boolean",
                      "description":"1"
                  },
                  "window_length":{
                      "type":"integer",
                      "minimum":15,
                      "maximum":300,
                      "description":"1"
                  },
                  "trigger_threshold":{
                      "type":"integer",
                      "minimum":1
                  },
                  "blocking_rate":{
                      "type":"number",
                      "minimum":0,
                      "maximum":100
                  }


              }
          }
      }
  },
  "notify_schema":{
      "type":"object",
      "additionalProperties":False,
      "required":["policy_type_id", "policy_instance_id", "handler_id", "status"],
      "properties":{
          "policy_type_id":{
              "type":"integer",
              "enum":[21000]
          },
          "policy_instance_id":{
              "type":"string"
          },
          "handler_id":{
              "type":"string"
          },
          "status":{
              "type":"string",
              "enum":["OK", "ERROR", "DELETED"]
          }
      }
  }
}
```