VET5G: A Virtual End-to-End Testbed for 5G Network Security Experimentation

Zhixin Wen, Harsh Sanjay Pacherkar, Guanhua Yan

Department of Computer Science
Binghamton University

This work is supported by NSF Award CNS-1943079
5G deployment has been ongoing around the world

Attacks and defenses on 5G are being discovered and proposed, however either no experiments were conducted or was conducted in 5G NSA mode
Existing 5G implementations/testbeds

So far open source/academic 5G solutions are mostly focused on either core network or air interface.

- Open5GS
- free5GC
- Open Air Interface
- SRS RAN

open5Gcore is closest to our work, however it is not designed for security research.
Testbeds

• DETER Project
• SCADA Testbed
• Cyber-Power Testbed
• Security Testbed for Internet-of-Things Devices
• Emulab
• And more...
Goal & Solution

To address the problem of lacking of testbeds, we introduce **VET5G** to meet the following requirements:

1. **End-to-end** emulation from UE to core network
2. **Easy** to use and **ready** to use testbed with JupyterLab

Doing so researchers can focus on research not testbed setup, saving time.

In addition we provide

3. **Secure** 5G core implementation in rust
4. **Programmable** UPF with P4
Architecture overall
Architecture overall

- Multiple PLMNs, each contains:
  - Core network
  - RAN
  - UEs
Core network emulation

- Core implemented in rust
- UPF customizable with P4 (BMv2)
Access Network emulation

- OpenAirinterface’s gNB is used
- Emulates layers all the way down to MAC
- UE and RAN communicates via nFAPI
User Equipment emulation

- Android emulator with Android 11 is used
- OAI’s nrUE is used to emulate baseband processor
User Interface
Orchestration & User Interface

- **Kubernetes** is used to orchestrate all containerized components
- Users interact with the testbed via **JupyterHub**
- Users define the testbed via yaml config file, this includes
  - Config of each container, be that NF, UE or gNB
  - Config of each subscriber

```
smfs:
- name: 'smf1'
  slices:
    - name: 'slice-1'
      sst: 1
      sd: '000001'

ip_pool:
- name: 'default'
  ipv4_cidr: '10.10.0.0/16'
  allowed_slices: null
dump_pcap: true
dump_logs: true
```

```
subscribers:
- imsi: '2689900007487'
  ul_ambr: 100 # unit: KBps
dl_ambr: 100 # unit: KBps
  subscribed_slices:
    - name: 'slice-1'
      sst: 1
      sd: '000001'
```

### Python API

<table>
<thead>
<tr>
<th>Python API</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_subscribers</td>
<td>Create mobile subscribers within a PLMN</td>
</tr>
<tr>
<td>start_core_network</td>
<td>Start a core network instance within a PLMN</td>
</tr>
<tr>
<td>start_gnb</td>
<td>Start gNB(s) within a PLMN</td>
</tr>
<tr>
<td>start_ue</td>
<td>Start UE(s) within a PLMN</td>
</tr>
<tr>
<td>execute_shell_command</td>
<td>Execute a shell command in an Android Emulator</td>
</tr>
<tr>
<td>install_apk</td>
<td>Install an Android APK in an Android Emulator and return its package name</td>
</tr>
<tr>
<td>start_app</td>
<td>Start an Android APK with a given package name in an Android Emulator</td>
</tr>
<tr>
<td>stop_all</td>
<td>Stop an experiment and collect results</td>
</tr>
<tr>
<td>get_nf_log</td>
<td>Get logs of an NF instance (it must be called after stop_all)</td>
</tr>
</tbody>
</table>
## Orchestration & User Interface

```python
from vet5g.client import VET5GClient
from vet5g.models import *

plmn = PlmnId("301", "91")
vet5g = VET5GClient("username", [plmn], "/tmp/"

await vet5g.create_subscribers(plmn, cfg_file = 'demo1_subscribers.yaml')
cn_cfg = await vet5g.start_core_network(plmn, cfg_file = 'demo1_cn.yaml')
gnb_cfg = await vet5g.start_gnb(plmn, cfg_file = 'demo1_gnb.yaml')
ue_cfg = await vet5g.start_ue(plmn, cfg_file = 'demo1_ue.yaml')
await vet5g.stop_all(pcap)
```

<table>
<thead>
<tr>
<th>Python API</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_subscribers</td>
<td>Create mobile subscribers within a PLMN</td>
</tr>
<tr>
<td>start_core_network</td>
<td>Start a core network instance within a PLMN</td>
</tr>
<tr>
<td>start_gnb</td>
<td>Start gNB(s) within a PLMN</td>
</tr>
<tr>
<td>start_ue</td>
<td>Start UE(s) within a PLMN</td>
</tr>
<tr>
<td>execute_shell_command</td>
<td>Execute a shell command in an Android Emulator</td>
</tr>
<tr>
<td>install_apk</td>
<td>Install an Android APK in an Android Emulator and return its package name</td>
</tr>
<tr>
<td>start_app</td>
<td>Start an Android APK with a given package name in an Android Emulator</td>
</tr>
<tr>
<td>stop_all</td>
<td>Stop an experiment and collect results</td>
</tr>
<tr>
<td>get_nf_log</td>
<td>Get logs of an NF instance (it must be called after stop_all)</td>
</tr>
</tbody>
</table>
VET5G use cases

• 1. Slicing attack
• 2. Cellular botnet and its defense
• 3. Course project
Slicing attack

- A design flaw in 5G protocol gives attacker access to information in another slice it otherwise shouldn’t
- We can simulate this attack by creating a custom NF using scaffolding code provided

```yaml
custom_nfs:
  - name: slicing-attacker
    image_name: 'slicing-attacker-nf.tar'
    image_args: []
    slices:
      - name: 'slice-2'
        sst: 1
        sd: '000002'
        allowed_slices: null
        dump_pcap: true
        dump_logs: true
```
Cellular botnet and its defense

- As more devices are connected, the harm a botnet can cause to mobile network increases
- Simple 5G botnet and its defense is demonstrated here
- Matryosh-like cellular botnet is studied here
- P4 **programmable UPF** allows defense to be quickly **prototyped**
Cellular botnet and its defense
Course project

• VET5G is here used for educational activity
• 40 students in 15 groups try to attack 5G system from an vantage point within the core network

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Description</th>
<th>Number of groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuzzing</td>
<td>Use tools such as wfuzz and sfuzz to fuzz test 5G core NFs</td>
<td>14</td>
</tr>
<tr>
<td>Reconnaissance</td>
<td>Use tools such as nmap to find attack targets in 5G networks</td>
<td>9</td>
</tr>
<tr>
<td>GTP attack</td>
<td>Guess TEIDs (Tunnel Endpoint Identifiers) used by legitimate GTP sessions</td>
<td>3</td>
</tr>
<tr>
<td>DDoS attack</td>
<td>Use tools such as slowloris to perform DDoS attacks</td>
<td>3</td>
</tr>
<tr>
<td>Password attack</td>
<td>Use tools such as Hydra to guess passwords</td>
<td>3</td>
</tr>
<tr>
<td>MITM attack</td>
<td>Use tools such as Bettercap to perform MITM (Man-In-The-Middle) attacks</td>
<td>2</td>
</tr>
<tr>
<td>SSH attack</td>
<td>Use tools such as metasploit to attack vulnerable SSH services</td>
<td>1</td>
</tr>
</tbody>
</table>
Course project

![Graph showing CPU and Memory Usage over days.](image)
Conclusion & Future work

- Integrate real SDR devices in VET5G we can use real 5G devices
- Incorporating more diverse end devices to study their security in a holistic way
- Integrate a Tofino based programable UPF
- Intentionally introduce vulnerabilities for future CTF events
- Study the security of rust implementation