Telecom Equipment Assurance Testing

T.V. Prabhakar,
Gopi Krishna S Garge,

Indian Institute of Science
Bangalore
Agenda

• Overview of the TETC
• Security Testing & requirements
• Security Standards?
• Is there a formalism to what we want?
• Can TTCN 3 help?
• Discussion
Our Mission

- Telecom space
  - Telecom includes data networking; focus on DN
  - Equipment acceptance tests
  - Security Evaluation
  - Safe-to-connect certification
  - Publish guidelines for procurers and OEMs
Objectives

• Set up an assurance test facility
• Tests include
  – Telecom Equipment (untrusted)
  – Detect hidden malicious code/systems within
  – Other h/w and s/w weaknesses that may exist
• Set up contractual terms for suppliers
• Review the requirements of such assurance facilities
Assurance Testing

• Product and System assurance
• Suite of tests
  – Vulnerability Analysis
  – Penetration Testing (BB and fuzzing)
  – Deep Inspection (source code, processes, etc.)
  – Non-functional tests, SVCT, MVCT, etc.
Assurance Framework

• Common criteria (adapted?)
  – Criteria, methodology and recognition for IT security evaluation
  – Protection Profiles
  – Security Targets
  – Testing and Evaluation

• Can we use TTCN 3 in such a context?
Security Evaluation

- Risk estimation and Deployment Targets
- What to protect?
- What protection to evaluate?
- Formal Representation? Grammar?
- Translate to a spec language?
- Derive test suites?
  - Code for execution
  - Code inclusion
  - Verdict/security level quantification
Security Tests

- Compliant vs Vulnerable
- Test Design
  - SUT
    - Load Conditions
    - Responses
    - Graceful degradation/recovery
  - Attack Parameters
    - Persistent vs non-persistent
    - low/med/high persistence
    - Single vs multiple attacks
    - Detection avoidance
TTCN-3 Applications

- Mobile communications
  - LTE, WiMAX, 3G, TETRA, GSM
- Broadband technologies
  - ATM, DSL
- Middleware platforms
  - WebServices, CORBA, CCM, EJB
- Internet protocols
  - SIP, IMS, IPv6 and SIGTRAN
- Smart Cards
- Automotive
  - AUTOSAR, MOST, CAN
Security Standards

- ETSI and the eEurope programme – 2005
- STF 356 – Making better security standards
- 4th ETSI Security Workshop
  - EG 202 387, Common Criteria
  - ES 202 382, Protection Profile
  - ES 202 383, Security Target
Security Standards

- Any requirement should be testable
- Any security requirement must be testable AND must achieve its security objective
- Open development of crypto has been the norm for a number of years (AES for example)
- Security systems need to be open to examination
- Assurance evaluation schemes fit the model
- Designing in anticipation of assurance evaluation is good practice
Security Standards

• Risk analysis is still top of the process tree
• Objectives still have to be established before requirements
• Crypto based solutions by themselves don’t provide security
Security Testing

• Telecom equipment security testing means:
  – Equipment is free from vulnerabilities
    • DOS, Buffer overflow, Remote Code Execution, Format string, Malloc bombs, ..
  – Equipment is free from virus and malware
  – Equipment is recommended for “safe to connect”
Security testing approaches

• Several approaches are possible:
  – Attack the equipment and observe its capability to withstand or mitigate the attack
    • Attack heuristics can be developed
  – Perform a black box robustness testing and look for implementation level security
    • Design test cases
  – Complete coverage of the input space
  – Monitor traffic with a sniffer and analyze the data with appropriate filters
  – Monitor a deviation from the baseline – anomaly detection?
Security testing

• TTCN-3 based security test suites, when done, have to be made publicly available

• Threat and Risk perceptions master script
  – Recommends the actual scripts that are required to be run

• Certification scripts adhering to security standards are urgently required
  – Common Criteria based Protection profiles will be invaluable

• Client-Server/ Peer scripts to maintain security assurance of production equipment
  – Eg: Impact of opening a firewall’s port on core router
Using TTCN 3

• Grammar for expressing network policy violations
• Representation of exploits as action sequence trees
• Compliance vs Vulnerability
• Stateful protocols
• Synchronization
The formalism available: An Illustration
Terminology
### Characteristic Grammar Definitions

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS-CONFIGURATION</td>
<td>Device or software configuration violates network policy</td>
</tr>
<tr>
<td>MIS-SPECIFICATION</td>
<td>Intended semantics not captured by protocol specification, example: mandating broken encryption schemes</td>
</tr>
<tr>
<td>FLAW</td>
<td>An unintended condition present in a system</td>
</tr>
<tr>
<td>SOFTWARE FLAW</td>
<td>Application or operating system violates protocol design</td>
</tr>
<tr>
<td>HARDWARE FLAW</td>
<td>Hardware violates protocol design</td>
</tr>
<tr>
<td>AUTHENTICATION FLAW</td>
<td>Protocol verifies identities insufficiently or not at all</td>
</tr>
<tr>
<td>AUTHORIZATION FLAW</td>
<td>Protocol verifies permissions insufficiently or not at all</td>
</tr>
<tr>
<td>NONCE FLAW</td>
<td>Protocol binds data to session non-uniquely or not at all</td>
</tr>
<tr>
<td>STATE MACHINE FLAW</td>
<td>Protocol specification tracks internal state insufficiently or not at all, enabling flooding and brute forcing</td>
</tr>
</tbody>
</table>

### Characteristic Grammar

```
<characteristic> ::= MIS-CONFIGURATION | MIS-SPECIFICATION | SOFTWARE FLAW | HARDWARE FLAW | AUTHENTICATION FLAW | AUTHORIZATION FLAW | NONCE FLAW | STATE MACHINE FLAW
```
# Symptoms

<table>
<thead>
<tr>
<th>Symptom Grammar Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>USER CREDENTIALS</strong></td>
</tr>
<tr>
<td><strong>USER DATA</strong></td>
</tr>
<tr>
<td><strong>HOST CREDENTIALS</strong></td>
</tr>
<tr>
<td><strong>HOST DATA</strong></td>
</tr>
<tr>
<td><strong>HOST SERVICE</strong></td>
</tr>
<tr>
<td><strong>CONNECTION</strong></td>
</tr>
<tr>
<td><strong>BANDWIDTH</strong></td>
</tr>
<tr>
<td><strong>DIVERT</strong></td>
</tr>
<tr>
<td><strong>DISABLE</strong></td>
</tr>
<tr>
<td><strong>SNIFFING</strong></td>
</tr>
<tr>
<td><strong>IMPERSONATING</strong></td>
</tr>
<tr>
<td><strong>CONNECTION HIJACKING</strong></td>
</tr>
<tr>
<td><strong>MAN IN THE MIDDLE</strong></td>
</tr>
<tr>
<td><strong>DENIAL OF SERVICE</strong></td>
</tr>
<tr>
<td><strong>FLOOD</strong></td>
</tr>
</tbody>
</table>
Symptom Definitions

Symptom Grammar

<symptom> ::= DIVERT <resource> | DISABLE <resource>

<resource> ::= <user resource> | <host resource> | <network resource>

<user resource> ::= USER CREDENTIALS | USER DATA

<host resource> ::= HOST CREDENTIALS | HOST DATA | HOST SERVICE

<network resource> ::= CONNECTION | BANDWIDTH

<sniifing> ::= DIVERT USER DATA

<impersonating> ::= DIVERT USER CREDENTIALS | DIVERT HOST CREDENTIALS

<connection hijacking> ::= DIVERT CONNECTION \ DIVERT HOST CREDENTIALS

<man in the middle> ::= <connection hijacking>+ \n
<flood> ::= DISABLE BANDWIDTH

<denial of service> ::= DISABLE HOST SERVICE | <man in the middle> | <flood>
Vulnerability

Vulnerability Grammar

\[
\begin{align*}
\text{<vulnerability>} & ::= \text{MIS-CONFIGURATION}^{+} \mid \text{<implementation>}^{+} \mid \text{<design>}^{+} \\
\text{<implementation>} & ::= \text{SOFTWARE FLAW} \mid \text{HARDWARE FLAW} \\
\text{<design>} & ::= \text{MIS-SPECIFICATION} \mid \text{STATE MACHINE FLAW} \mid \text{<forgery>} \\
\text{<forgery>} & ::= \text{AUTHENTICATION FLAW} \mid \text{AUTHORIZATION FLAW} \mid \text{NONCE FLAW}
\end{align*}
\]
Exploit Grammar

<exploit> ::= <vulnerability>+ <symptom>+
Algorithm

Is-SECURE (Protocol $P$, Implementation $I$, Policy $\rho$)

1. $secure \leftarrow \neg \text{Is-MisConfigured} (I, \rho)$
2. $secure \leftarrow secure \land \neg \text{Is-MisSpecified} (P)$
3. $secure \leftarrow secure \land \neg \text{Has-Implementation-Flaw} (I)$
4. $secure \leftarrow secure \land \neg \text{Has-State-Machine-Flaw} (P)$
5. for each packet type $p \in P$
   6. \hspace{1em} $secure_p \leftarrow \neg \text{Has-Authentication-Flaw} (p)$
   7. \hspace{1em} $secure_p \leftarrow secure_p \land \neg \text{Has-Authorization-Flaw} (p)$
   8. \hspace{1em} $secure_p \leftarrow secure_p \land \neg \text{Has-Nonce-Flaw} (p)$
   9. \hspace{1em} $secure_p \leftarrow secure_p \lor \text{Allows-Forgery} (p)$
10. \hspace{1em} secure \leftarrow secure \land secure_p$
11. return $secure$
ntp Vulnerability

exploit

vulnerability

design-flaw

forger

authentication-flaw

key index

denial-of-service

VLAN Vulnerability

Diagram:
- Exploit
  - Vulnerability
  - Design-flaw
    - Forgery
      - Authentication-flaw
        - Server ads without md5
  - Design-flaw
    - Forgery
      - Nonce-flaw
        - Server ads with md5
  - Symptom
    - Denial-of-service
So,

- Is this formalism helpful?
- What is required in terms of functions and libraries?
- Use the IPv6 core and common libraries to generate prototype test suites?
- Follow up with a similar approach for layer 4 protocols? Is this feasible?
- Known effort - TTCN 3 and Security – T3FAH
Thank You