Risk Assessment of Smart Home Security

Name: Steven Matuvu
1st Supervisor: Maryam Shapasand
2nd Supervisor: Samuel Onalo

STAFFORDSHIRE UNIVERSITY
LONDON DIGITAL INSTITUTE

GRAD EX
25TH ANNIVERSARY
3 - 7 JUNE, 2024
UNLIMITED CONNECTIONS

Introduction:

The project aims to assess the risks in the security of smart homes, in terms of security vulnerabilities that are persistent in Smart Home Networks

Research Gap:

Lack of Security Research based on the Integration of various devices from different manufacturers.

Demand for Risk Assessment frameworks which can adapt to the continuous change in Smart Home technology

All of the existing scoring systems provide scores manually

Importance of Addressing the Gap:

Enhanced Security and Protection

More Awareness on existing vulnerabilities

Future proofing against emerging threats

Solution:

Automated Vulnerability Scoring System

Scans the network for vulnerabilities
Then provides an overall score based on the vulnerabilities found.

Key Features:

-Network Scanning -CVSS Integration
-CVE Integration -Vulnerability Scoring

Technologies Used:

Nmap
Python 3

CVSS (Scoring System)
CVE (Vulnerability Database)

NMAP.ORG

Interface type:

The solution uses a command line interface CLI

```
Free Pascal Compiler version 3.1.1-r[illegible] [2018/[illegible]] for x86_64
Copyright (c) 1993-2018 by Florian Klaempfl and others
[illegible]\fpc.exe [options] <inputfile> [options]
 Only options valid for the default or selected platform are listed.
 Put + after a boolean switch option to enable it, - to disable it.
  @<x>   Read compiler options from <x> in addition to the default fpc.cfg
  -a     The compiler does not delete the generated assembler file
      -a5        Don't generate Big Obj COFF files for GNU Binutils older than 2.25 (Windows, NativeNT)
      -al        List sourcecode lines in assembler file
      -an        List node info in assembler file (-dEXTDEBUG compiler)
      -ao        Add an extra option to external assembler call (ignored for internal)
      -ar        List register allocation/release info in assembler file
      -at        List temp allocation/release info in assembler file
  -A<x>  Output format:
      -Adefault  Use default assembler
      -Aas       Assemble using GNU AS
      -Agas      Assemble using GNU GAS
      -Agas-darwin Assemble darwin Mach-O64 using GNU GAS
      -Amasm     Win64 object file using ml64 (Microsoft)
      -Apecoff   PE-COFF (Win64) using internal writer
      -Aelf      ELF (Linux-64bit) using internal writer
      -Ayasm     Assemble using Yasm (experimental)
      -Anasm     Assemble using Nasm (experimental)
      -Anasmwin64 Assemble Win64 object file using Nasm (experimental)
      -Anasmelf  Assemble Linux-64bit object file using Nasm (experimental)
      -Anasmdarwin Assemble darwin macho64 object file using Nasm (experimental)
  -b     Generate browser info
      -bl        Generate local symbol info
  -B     Build all modules
  -C<x>  Code generation options:
```

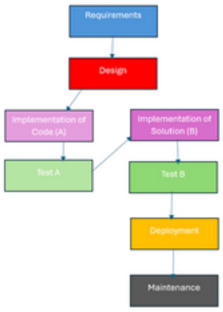
Methodology

Waterfall Methodology:

The solution was managed using the Waterfall methodology

Implementation was split into two stages.

A – Code
B – Complete Solution



MoSCoW:

This methodology was used for outlining requirements and testing

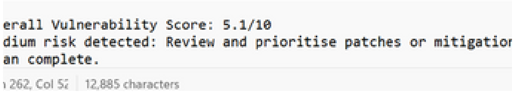
Requirements are organised under 4 sections:

Must have
Should have
Could have
Won't have

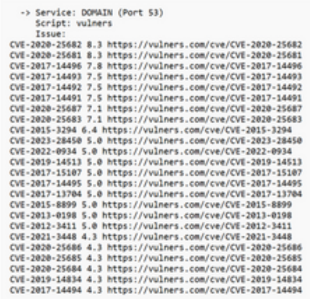
Testing and Results

The Testing was split into two sections like illustrated in the methodology. Two tests were ran:

Test A – The Code
Test B – The complete solution



```
erall Vulnerability Score: 5.1/10
dium risk detected: Review and prioritise patches or mitigatior
an complete.
```



```
  -> Service: DOMAIN (Port 53)
     Script: vulners
     Issue:
CVE-2020-25682 8.3 https://vulners.com/cve/CVE-2020-25682
CVE-2020-25681 8.3 https://vulners.com/cve/CVE-2020-25681
CVE-2017-14496 7.8 https://vulners.com/cve/CVE-2017-14496
CVE-2017-14493 7.5 https://vulners.com/cve/CVE-2017-14493
CVE-2017-14492 7.5 https://vulners.com/cve/CVE-2017-14492
CVE-2017-14491 7.5 https://vulners.com/cve/CVE-2017-14491
CVE-2020-25687 7.1 https://vulners.com/cve/CVE-2020-25687
CVE-2020-25683 7.1 https://vulners.com/cve/CVE-2020-25683
CVE-2015-3294 6.4 https://vulners.com/cve/CVE-2015-3294
CVE-2023-28450 5.0 https://vulners.com/cve/CVE-2023-28450
CVE-2022-0934 5.0 https://vulners.com/cve/CVE-2022-0934
CVE-2019-14513 5.0 https://vulners.com/cve/CVE-2019-14513
CVE-2017-15107 5.0 https://vulners.com/cve/CVE-2017-15107
CVE-2017-14495 5.0 https://vulners.com/cve/CVE-2017-14495
CVE-2017-13704 5.0 https://vulners.com/cve/CVE-2017-13704
CVE-2015-8899 5.0 https://vulners.com/cve/CVE-2015-8899
CVE-2013-0198 5.0 https://vulners.com/cve/CVE-2013-0198
CVE-2012-3411 5.0 https://vulners.com/cve/CVE-2012-3411
CVE-2021-3448 4.3 https://vulners.com/cve/CVE-2021-3448
CVE-2020-25686 4.3 https://vulners.com/cve/CVE-2020-25686
CVE-2020-25685 4.3 https://vulners.com/cve/CVE-2020-25685
CVE-2020-25684 4.3 https://vulners.com/cve/CVE-2020-25684
CVE-2019-14834 4.3 https://vulners.com/cve/CVE-2019-14834
CVE-2017-14494 4.3 https://vulners.com/cve/CVE-2017-14494
```

Evaluation

Findings:

- Security flaws in Smart Home Systems
- Lack of security standards
- Dependencies of Smart Homes
- Lack of risk assessment material for smart homes

Contributions:

- Security awareness
- Solution which addresses weaknesses
- More compliance

Future Works:

- Research into integrating machine learning
- Mitigation strategies for vulnerabilities found
- Improvements to Smart Home Security

This project is only the starting point!