AMI Penetration Test Plan

Version 1.0

Primary Author:
Justin Searle, Utilisec

Contributers:
Galen Rasche, EPRI
Andrew Wright, N-Dimension Solutions
Scott Dinnage, N-Dimension Solutions

Reviewers:
NESCOR Team 3 Members and Volunteers
Annabelle Lee, EPRI

Introduction

This security test plan template was created by the National Electric Sector Cybersecurity Organization Resource (NESCOR) to provide guidance to electric utilities on how to perform penetration tests on AMI systems. Penetration testing is one of the many different types of assessments utilities can perform to assess their overall security posture. While NESCOR recommends that utilities engage in all other forms of security assessment, NESCOR created this document to help utilities plan and organize their AMI penetration testing efforts. For a list of other types of Smart Grid security assessments, please see NESCOR's whitepaper titled "Guide to Smart Grid Assessments." For a list of other NESCOR Penetration Test Plan documents that cover other systems such as Wide-Area Monitoring, Protection, and Control (WAMPAC), Home Area Network (HAN), or Distribution Management, please see NESCOR's website or contact one of the persons listed above.

The objective of the NESCOR project is to establish an organization that has the knowledge and capacity to enhance the effort of the National Electric Sector Cybersecurity Organization (NESCO) by providing technical assessments of power system and cybersecurity standards to meet power system security requirements; provide recommendations for threats and vulnerabilities, and participate in testing emerging security technologies in labs and pilot projects.

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NESCOR, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

Table of Contents

1 Constraints and Assumptions

This document was created for electric utilities to use in their Smart Grid security assessment of AMI systems. Smart Grid security assessments can be broken into several categories. This document focuses only on penetration testing and attempts to help utilities break down the complex process of penetration testing. Penetration testing is a specialized form of hands-on assessment where the testing team takes on the role of the attacker and tries to find and exploit vulnerabilities in systems and devices. Testers use the same methodology that attackers use to identify vulnerabilities in a system. Once a vulnerability is found, the testers attempt to exploit the flaw to gain a foothold in the system and begin the process again to discover additional, lower level vulnerabilities that weren't previously exposed. Penetration testing is distinguished from vulnerability assessment techniques by the fact that they test for a depth of vulnerabilities instead of simply breadth, focus on discovering both known and unknown vulnerabilities, and provide the testing team with a better understanding of a particular vulnerability's risk to the business.

This document is intended to help electric utility security teams plan their penetration testing activities and understand rough levels of effort they should expect when performing these types of tests. When electric utilities do not have staff with the appropriate understanding or skill to perform penetration testing in-house, this document can be used in their services procurement processes to create RFP documents and evaluate the responses from potential firms offering penetration-testing services.

This document breaks the process of penetration testing into logical tasks. These tasks are organized into logical sections based on the skill set of the testing team. Not all penetration testers have the skill set to perform all of the tasks. In most cases, the testing team will be made up of at least two individuals, each with unique but (hopefully) somewhat overlapping skill sets. Because of the nature of penetration testing, the tasks in this document are high level and intended to break the overall penetration test into logical components that can be assigned to testing team members to be completed in a systematic manner. **This document does not contain detailed, tool specific, step-by-step procedures for each task, but provides high-level descriptions of how a task is performed and the overall goals for each task in an AMI Penetration Test.**

Results of penetration testing tasks are not expected to be fully repeatable or comparable from one utility to another utility, or from one testing team to another testing team. While all vulnerabilities found by the penetration testing team should be repeatable or verifiable by other organizations, the results of penetration testing is highly dependent on the skill set of the testing team, and the discovery of those vulnerabilities will vary from testing team to testing team. Because of these factors, the results of these penetration-testing tasks are not intended to be used by regulatory bodies or shared outside of the utility, with the exception of sharing these results with the respective vendors to have the discovered vulnerabilities addressed.

2 Penetration Test Planning

Penetration testing should be performed on a periodic basis depending on the criticality of the targeted system. NESCOR recommends performing this type of assessment on an annual basis or after any major systems upgrades or changes.

Penetration tests should start with an architecture review to help the testing team gain a deeper knowledge of the target system. This will help the penetration testing team understand the intended functionality of the targeted system, its theoretical security posture from an architectural perspective, and the security risks that a vulnerability could pose to the organization.

Actual penetration tests should be performed on non-production systems and devices that are installed and configured for actual operation in testing or staging environments. The closer the target systems are configured to their production counterparts, the more accurate an assessment you will receive. This includes interconnectivity to dependent systems communicating with the targeted systems, such as the presence of a meter data management system (MDMS) connected to an AMI headend being testing. In cases where testing and staging environments do not exist, the testing team could select non-intrusive, low-risk penetration-testing tasks that can be done on production systems. NESCOR will not give guidance on which tasks are low-risk; this can only be determined by the testing team familiar with the target system. The nature of penetration testing is a trial and error method, often with unforeseen consequences in the systems being tested. Utilities would be wise to invest in testing or staging environments if they do not currently exist.

Each penetration-testing task listed in this document contains an estimated level of effort, a task description, and a task goal. The level of effort for each task assumes a single target. For example, if a task involves analyzing dataset for cryptographic keys and is labeled “medium” effort, this signifies that the analysis of each distinct dataset should be calculated as a separate medium level effort. The analysis of multiple datasets could aggregate to a “medium” or “high” level of effort depending on the exact relative nature of those datasets.

The following table was used to estimate the number of hours an **experienced tester** of the applicable skill set would take to complete each task:

Low Level of Effort	1-4 hours
Medium Level of Effort	5-16 hours
High Level of Effort	17-40 hours
Extremely High Level of Effort	41+ hours

The penetration-testing tasks included in this document were created to be used generically on all types of AMI systems. Therefore, individual penetration-testing tasks may or may not apply depending on the specific system being tested. The testing team that is performing the tasks should determine which tests are applicable to accomplish their goals.

Utilities should consider mapping their specific security requirements to each penetration-testing task to provide traceability and determine the value of each task. Based on these results, the utility may choose which penetration-testing tasks they pursue and which tasks they discard. They may also wish to place either financial or time-based constraints on the testing tasks to make sure they receive they expected cost-benefit ratio.

Figure 1 demonstrates how the following sections of this document interrelate to each other and when they are initiated in a typical penetration test. This diagram shows the overall process flow of a typical penetration test as described in this document. Each box represents a major section in this document and shows which sections need to be performed in serial and which sections can be performed in parallel.

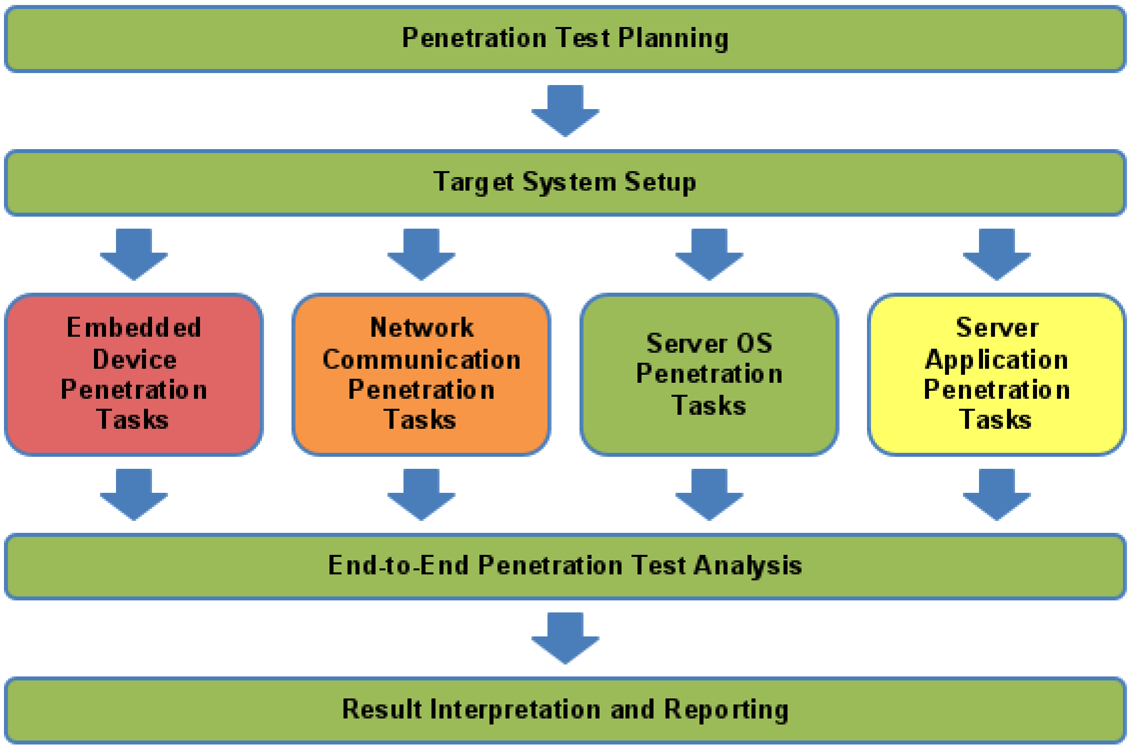

Penetration Test Planning
Target System Setup
Embedded Device Penetration Tasks
Network Communication Penetration Tasks
Server OS Penetration Tasks
Server Application Penetration Tasks
End-to-End Penetration Test Analysis
Result Interpretation and Reporting

Figure 1: Typical Penetration Testing Process

All penetration tests should start with proper planning and scoping of the engagement. Once that is complete, the penetration testing tasks can be broken into the four distinct categories displayed in Figure 1. Each of these task categories also requires different skill sets from the testing team. If there is sufficient staff, these four penetration task categories can be performed in parallel. Once these tasks are completed, the team should perform a gap analysis to verify all desired tests have been performed and all goals met. Finally, the team should generate a report documenting their findings, interpret these findings in the context of the utility's deployment, and develop recommendations to resolve or mitigate these vulnerabilities.

The color difference of these four penetration task categories represents the relative likelihood that a utility should consider performing these tasks. These recommendations are based a combination of trends that NESCOR has seen in the industry and the level of expertise needed to perform these tests. To some degree, this also represents the relative risk target systems represent to the utility, as compromise of the control servers are generally considered a higher risk than the compromise of a single embedded field device or its network communications.

The colors can be interpreted as:

- Green: Tasks that should be performed most frequently, require the most basic of penetration testing skill, and can often be performed by internal security teams.
- Yellow: Tasks that are commonly performed and require moderate penetration testing skill.
- Orange: Tasks that are occasionally performed but may require higher levels of expertise.
- Red: Tasks that are infrequently performed and require highly specialized skills not often found in-house.

These colors will also be used in the diagrams presented in each major task category in this document.

3 Target System Setup

The AMI systems to-be-tested should be configured for normal, expected operation in staging or test environments. This includes all components from the meter to the headend, and if in-scope, other control servers such as MDMS or Customer Information Systems (CIS) that may communicate with the headend system or any other AMI component. At a minimum, this document assumes functional communication from the meter to the headend, and this has been established before the penetration test begins. Furthermore, it is assumed that the testers have physical access to all devices in the test environment to perform penetration tasks.

AMI systems have been architected in a variety of different approaches. Figure 2 depicts a number of the most common architectures, including intermediate devices and possible communication links between the meter and the headend. This diagram attempts to include all major architecture types commonly deployed, however this means only a portion of this diagram may pertain to a specific utility. Therefore, this common architecture should be customized and tailored for specific AMI systems depending on the deployed devices and communication protocols.

Testers should be familiar with existing communication protocols that pass among different components within AMI infrastructures. Figure 3 depicts generic dataflows most AMI systems use in their communications between the headend and each meter.

Each one of the generic dataflows listed in Figure 3 represents a system functionality that attackers may leverage in their attacks. Testers should familiarize themselves with the administrative interface to the functionalities on both the meter and the headend sides. This knowledge will greatly aid testers during actual testing and enable them to trigger certain events when needed, such as initiating a firmware update while attempting to capture the update in one of the penetration test tasks.

Penetration testing tools play a key role in the testing process. Depending on the AMI component being testing, tools may not exist for each task. For example, at the time of writing, there were very few tools available to aid testers in the generation of common AMI communication protocols such as C12.18 (for optical communications on the meter) and C12.22 for meter-to-headend communication. If time and tester skill set permit, the tester can develop these tools as part of the testing. The level of effort for such tool development should be scoped as High (17-40 hours) or Extremely High (40+ hours).

Each penetration task category in this document will list the types of tools needed for the tasks in that category. This list should not be considered prescriptive or complete. The tools needed will vary between individual testers and will change over time.

Figure 2: Common AMI Architecture

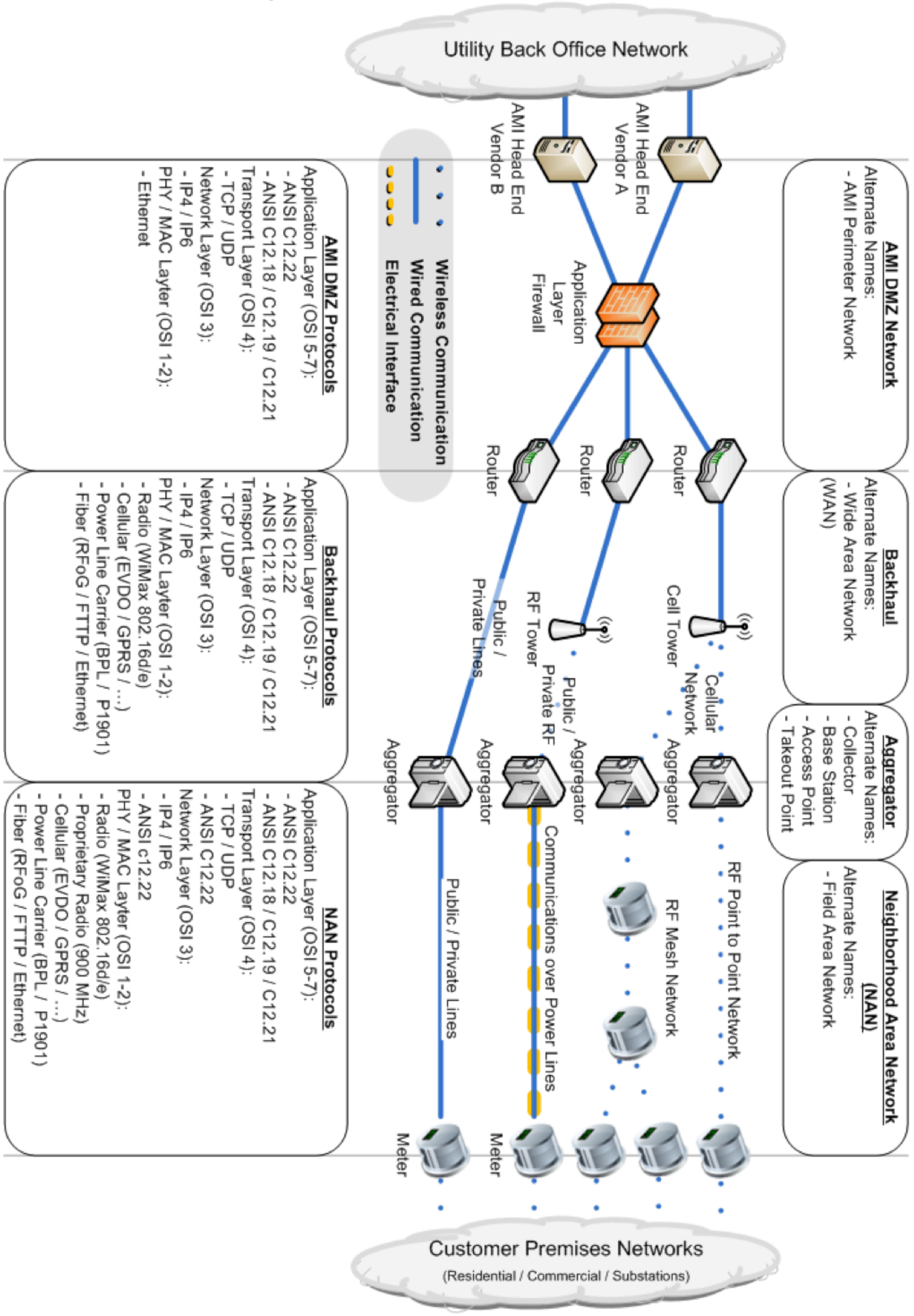
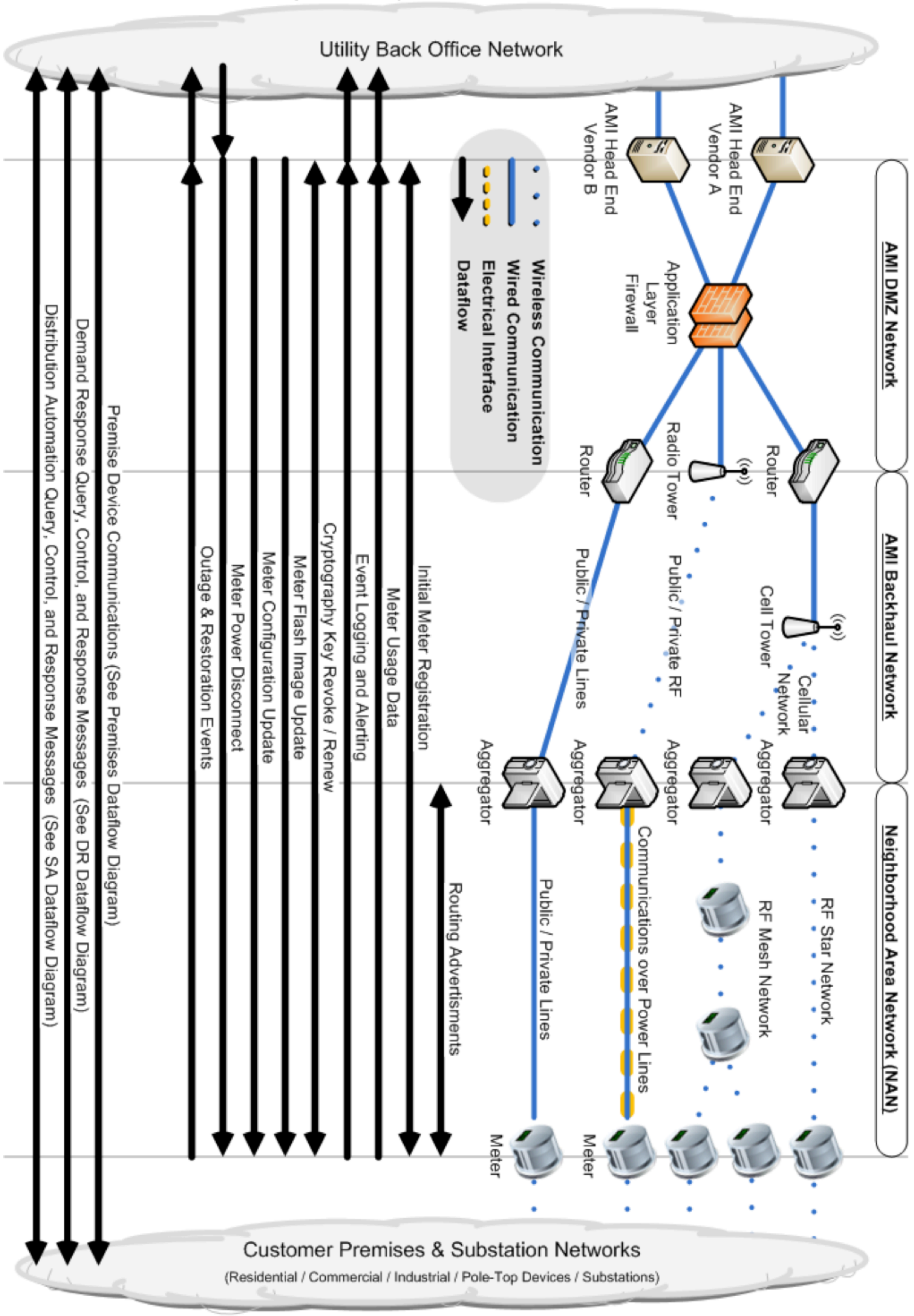


Figure 3: Typical AMI Dataflows


Utility Back Office Network
AMI DMZ Network
AMI Backhaul Network
Neighborhood Area Network (NAN)
AMI Head End Vendor A
AMI Head End Vendor B
Application Layer Firewall
Router
Radio Tower
Router
Cell Tower
Cellular Network
Public / Private RF
Public / Private Lines
Aggregator
Aggregator
Aggregator
Aggregator
RF Star Network
RF Mesh Network
Communications over Power Lines
Public / Private Lines
Meter
Meter
Wireless Communication
Wired Communication
Electrical Interface
Dataflow
Routing Advertisements
Initial Meter Registration
Meter Usage Data
Event Logging and Alerting
Cryptography Key Revoke / Renew
Meter Flash Image Update
Meter Configuration Update
Meter Power Disconnect
Outage & Restoration Events
Premise Device Communications (See Premises Dataflow Diagram)
Demand Response Query, Control, and Response Messages (See DR Dataflow Diagram)
Distribution Automation Query, Control, and Response Messages (See SA Dataflow Diagram)
Customer Premises & Substation Networks
(Residential / Commercial / Industrial / Pole-Top Devices / Substations)

4 Embedded Device Penetration Tasks

This section addresses the testing of field-deployed, embedded, microprocessor based devices. Hardware that is commonly deployed in areas where attackers could easily gain physical access should be tested using the tasks listed below. In AMI systems, this is typically the meter, relays, and aggregators (also known as access points in some architectures).

Primary targets for these tasks are the electronic components used inside the field devices. These tasks target electronic components that store data (EEPROM, Flash, RAM, MCu on-chip storage), buses that pass data between components (parallel buses and serial buses), and input interfaces used for administrative or debugging purposes (serial ports, parallel ports, infrared/optical ports). The overarching goal for embedded device testing is to identify vulnerabilities that allow attackers to expand their control of that single device to other devices with limited or no physical access to those other devices. For example, successful retrieval of an AMI meter's C12.18 master password, a password that protects the optical interface on the front of a meter, will enable attackers to directly interface with the optical port other meters without having to disconnect or dismantle the other meters. Of course this assumes that the master password is used throughout the smart meter deployment, which unfortunately is often the case.

Figure 4 below shows the overall process flow of the task sub-categories in this section. The figure shows the three task sub-categories may be performed in parallel. As in previous diagrams in this document, the colors represent the recommended likelihood that a utility should consider performing these task sub-categories, and the relative level of expertise required.

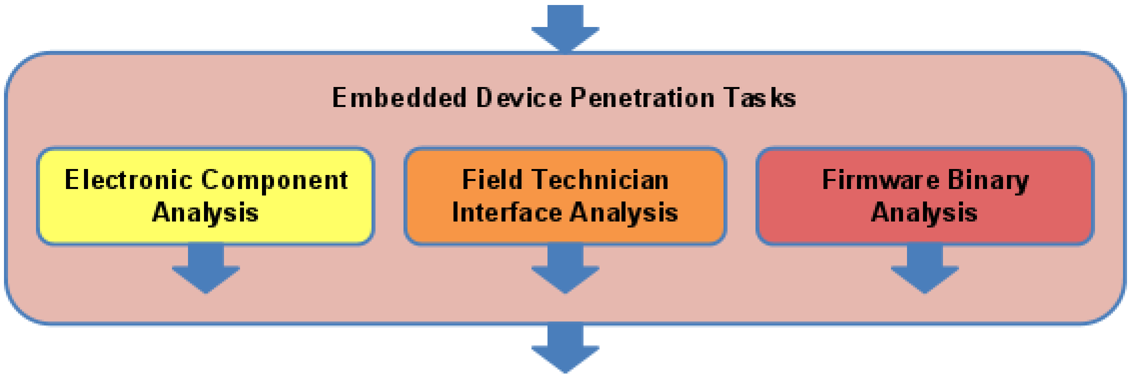

Embedded Device Penetration Tasks
Electronic Component Analysis
Field Technician Interface Analysis
Firmware Binary Analysis

Figure 4: Embedded Device Subcategory Flow

Each subcategory below includes a similar diagram depicting the process flow and recommended likelihood to perform each task.

Suggested Tools:

- Basic tools such as screw drivers, wire cutters, pliers, tin snips, etc.
- Electronics equipment such as power supply, digital multimeter, and oscilloscope
- Electronic prototyping supplies such as breadboard, wires, components, alligator jumpers, etc.
- Specialized tools to communicate directly with individual chips or capture serial communications such as a Bus Pirate or commercial equivalent such as Total Phase Aardvark/Beagle.
- Universal JTAG tool such as a GoodFET
- Surface mount micro test clips
- Electric meter test socket
- Disassembler Software for the appropriate microprocessors to be tested
- Entropy Analysis Software
- Protocol Analysis Software

4.1 Electronic Component Analysis

This subcategory of penetration tasks focuses on the identification design weaknesses in the electronic components. Often these weaknesses show themselves in unprotected storage or transfer of sensitive information such as cryptographic keys, firmware, and any other information that an attacker can leverage to expand his attack. In AMI systems, this usually equates to C12.18 passwords for optical ports, any cryptographic keys used in communications with other devices (C12.21, C12.22, or other protocols the embedded field device uses), and any firmware the device may use (usually one per microprocessor). Figure 5 shows a typical task flow for analyzing electronic components.

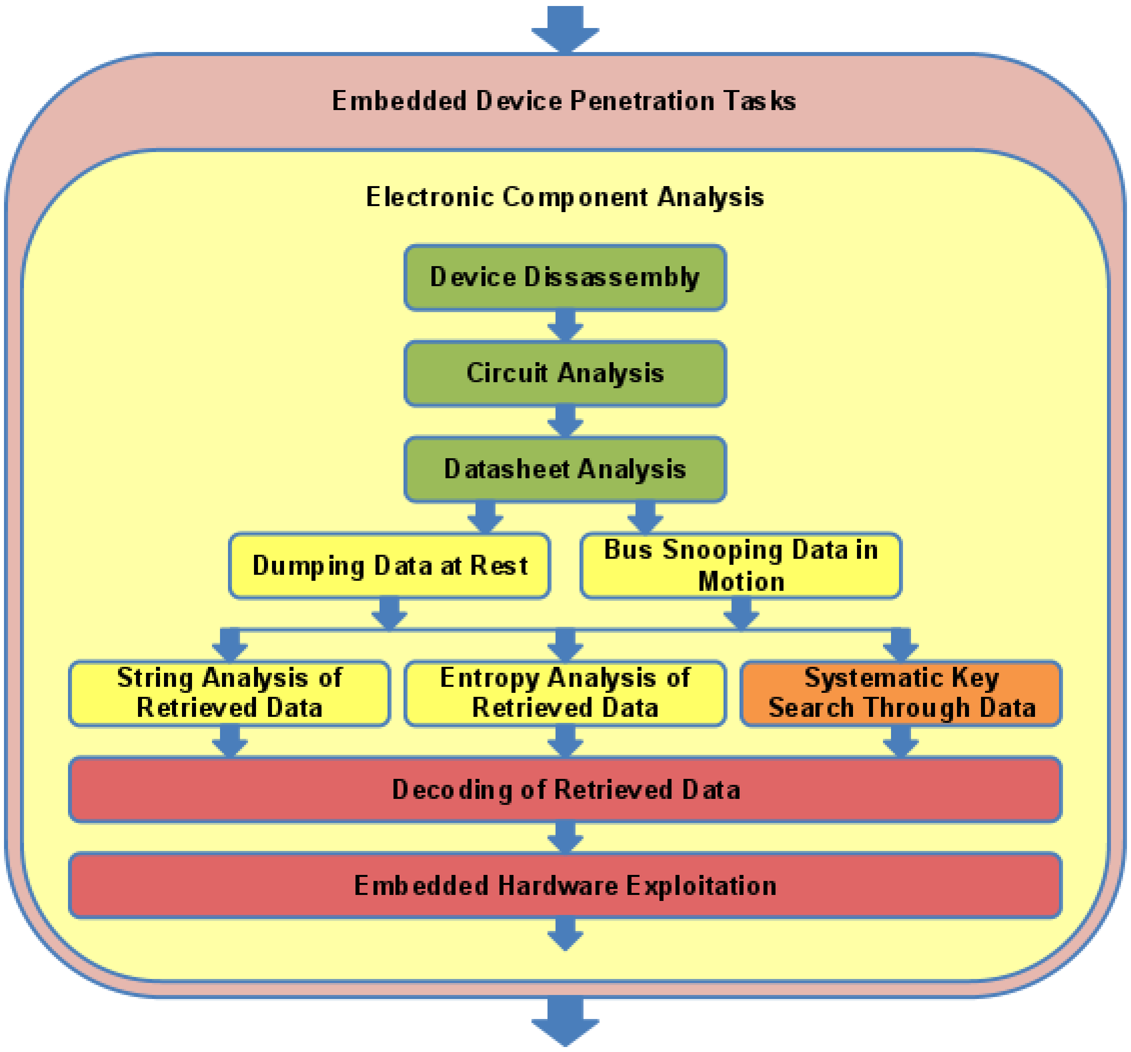

Embedded Device Penetration Tasks
Electronic Component Analysis
Device Dissassembly
Circuit Analysis
Datasheet Analysis
Dumping Data at Rest
Bus Snooping Data in Motion
String Analysis of Retrieved Data
Entropy Analysis of Retrieved Data
Systematic Key Search Through Data
Decoding of Retrieved Data
Embedded Hardware Exploitation

Figure 5: Electronic Component Analysis Task Flow

4.1.1 Device Disassembly

Level of Effort: Low

Task Description: Disconnect power from the device and disassemble the device to gain access to the embedded electronic components. Attempt to do a non-destructive disassembly if possible. Document the entire process to later facilitate reassembly. Identify the existence and function of any physical tamper mechanisms protecting the device.

Task Goal: Gain physical access to embedded components and electronic buses for further testing. Identify any methods that could be used to bypass the tamper mechanisms..

4.1.2 Circuit Analysis

Level of Effort: Low

Task Description: Document the electronic circuit by taking pictures, reading chip IDs, tracing buses, and identifying major electronic functionality.

Task Goal: Gain information about the embedded hardware and identify potential electronic components for attack.

4.1.3 Datasheet Analysis

Level of Effort: Medium

Task Description: Find, download, and analyze all pertinent datasheets and related documentation for each major electronic component inside the device, to identify possible security weaknesses and attack angles.

Task Goal: Gain information about the function of each component and how to interface directly with each component. Identify target components and buses for following tasks.

4.1.4 Dumping Embedded Circuit Data at Rest

Level of Effort: Medium

Task Description: Using the datasheets, identify the pins necessary to perform data dumping. With the device powered off, connect your testing tools and perform the dump. If needed, be sure to disable any other component by triggering reset pins or by using other methods. Review the dumped data to determine if you were successful. Attempt multiple dumps and compare the results if you are doubtful about your success.

Task Goal: Obtain all data from unprotected storage devices for later analysis.

4.1.5 Bus Snooping Embedded Circuit Data in Motion

Level of Effort: Medium

Task Description: Using the datasheets previously obtained, identify the pins and traces needed to perform bus snooping. With the device powered off, connect the testing tools and begin capture. Power on the device and capture sufficient data samples from each target bus. Review dumped data to identify if you were successful. Attempt multiple dumps and compare results if you are doubtful about your success.

Task Goal: Obtain data samples from all major buses for later analysis.

4.1.6 String Analysis of Retrieved Data

Level of Effort: Low

Task Description: Use tools and multiple decoding methods to decode each obtained data. Within the logical context of the data source, identify human readable strings and other anomalies. Other identifiers may be byte patterns signifying where firmware image files begin and end.

Task Goal: Identify symmetric cryptographic keys, firmware images, and other items of interest.

4.1.7 Entropy Analysis of Retrieved Data

Level of Effort: Low to Medium

Task Description: Analyze obtained data sets for blocks of data that portray high levels of entropy. Small data blocks with high entropy often signify asymmetric cryptographic keys and usually correspond to common key length sizes. Larger data blocks with high levels of entropy often signify encrypted data. Attempt to use suspected cryptographic keys to decrypt encrypted data blocks or encrypted communications traffic.

Task Goal: Identify asymmetric cryptographic keys and encrypted data objects.

4.1.8 Systematic Key Search Through Data Sets

Level of Effort: Low

Task Description: Use tools to identify cryptographic keys by attempting to use possible blocks of data from each obtained data set as the cryptographic key.
For instance, if the tool is trying to identify a 128 bit symmetric key, the tool will systematically attempt to use each 128 bit data block to decrypt a known block of encrypted data or a known capture of encrypted communications traffic. The tool will try bits 0 through 127 as they cryptographic key, then try bits 1 through 128, then bits 2 through 129, etc.

Task Goal: Identify symmetric and asymmetric cryptographic keys.

4.1.9 Decoding of Retrieved Data

Level of Effort: High

Task Description: Reverse engineering of the data in an attempt to understand its purpose.

Task Goal: Identify the purpose of blocks of data that could be used in exploitation attempts.

4.1.10 Embedded Hardware Exploitation

Level of Effort: High to Extremely High

Task Description: Based on the findings from previous tasks, determine feasible attacks which can be launched on the embedded components.

Task Goal: Create proof of concept attacks to demonstrate the feasibility and business risk created by the discovered vulnerabilities.

4.2 Field Technician Interface Analysis

Most embedded devices provide physical interfaces for local configuration and debugging. In AMI field devices this is often an infrared optical port using the C12.18 protocol for communications. In non-meter devices, this may be a RS-232 or other serial interface. This subcategory of penetration tasks focuses on the analysis and identification of vulnerabilities in these interfaces. Figure 6 shows a typical task flow for testing field technician interfaces.

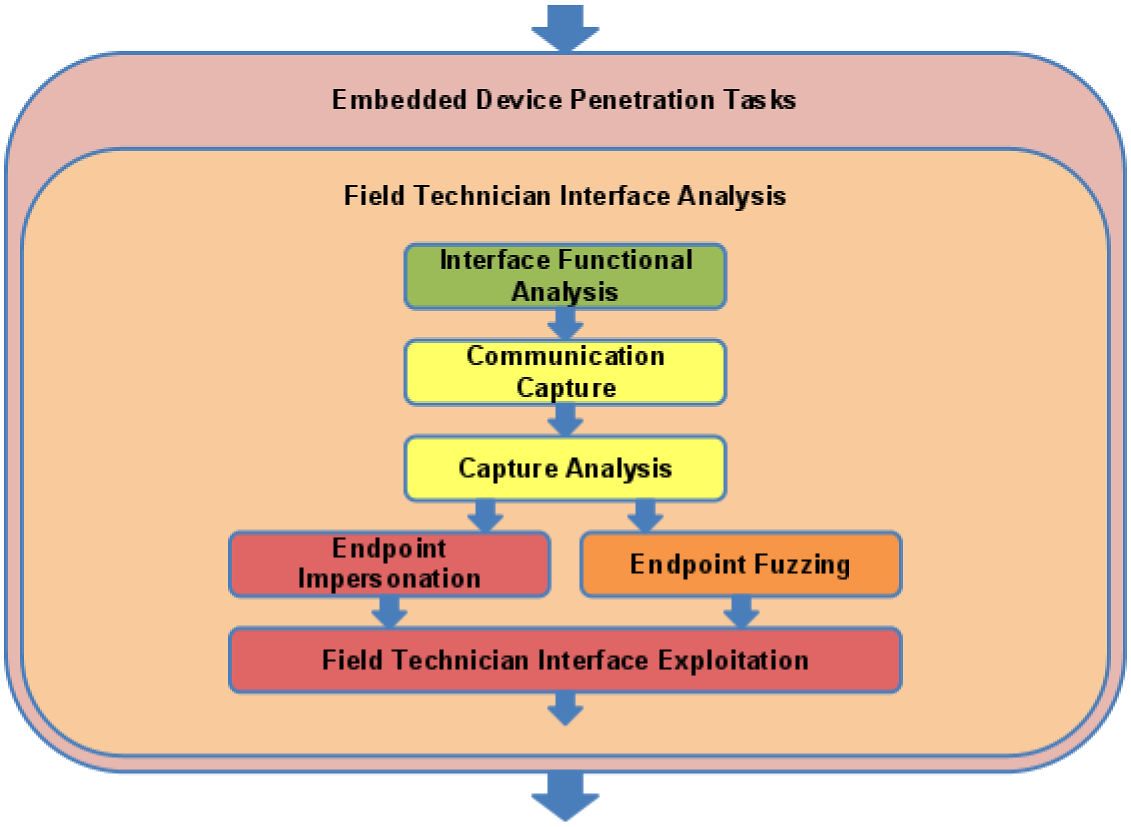

Embedded Device Penetration Tasks
Field Technician Interface Analysis
Interface Functional Analysis
Communication Capture
Capture Analysis
Endpoint Impersonation
Endpoint Fuzzing
Field Technician Interface Exploitation

Figure 6: Field Technician Device Task Flow

4.2.1 Interface Functional Analysis

Level of Effort: Low

Task Description: Obtain required software and hardware to establish an appropriate connection to the field device, be it a serial port, infrared port, or digital display. Identify the intended functionality and features of the interface. Identify any unprotected or high-risk functions that attackers may be interested in exploiting, such as firmware updates or security table reads.

Task Goal: Gain an understanding of the interface feature set and identify functions that should be targeted for later tasks.

4.2.2 Field Technician Interface Communications Capture

Level of Effort: Low

Task Description: Use a hardware or software tool to intercept normal communications on the interface. Capture all identified target functions from previous tasks.

Task Goal: Obtain low-level capture of targeted functions.

4.2.3 Field Technician Interface Capture Analysis

Level of Effort: Medium

Task Description: Analyze interface captures, identifying weaknesses in authentication, authorization, and integrity controls. Gain an understanding of how data is requested and commands are sent. If the protocol is the C12.18 protocol, attempt to identify the system passwords being sent in the clear for different types of commands.

Task Goal: Identify potential vulnerabilities and attacks.

4.2.4 Field Technician Interface Endpoint Impersonation

Level of Effort: Low to Medium

Task Description: Use a tool to impersonate either end of the field technician interface. For instance, this tool could simulate the field technician tool while communicating with the meter, or the tool could simulate the meter while communicating to with the meter.

Task Goal: Obtain a usable interface to perform tasks such as Interface Fuzzing.

4.2.5 Field Technician Interface Fuzzing

Level of Effort: Medium to High

Task Description: Use a tool to send both valid and invalid communications to the target interface, analyzing the results and identifying anomalies. This task includes items such as password guessing, invalid input testing, data enumeration, etc.

Task Goal: Identify vulnerabilities in the interface implementation and obtain data not otherwise available from any meter vendor tool provided to the utility.

4.2.6 Field Technician Interface Exploitation

Level of Effort: High to Extremely High

Task Description: Based on the findings from previous tasks determine feasible attacks which can be launched on the field technician interface.

Task Goal: Create proof of concept attacks to demonstrate the feasibility and business risk created by the discovered vulnerabilities.

4.3 Firmware Binary Analysis

This subcategory of penetration tasks focuses on the identification of vulnerabilities in binary firmware. These tasks do not describe traditional software source code review, rather they describe the techniques that attackers would use when they gain access to a firmware image in binary format and do not have access to the firmware's original source code. Binary analysis is very time intensive and could be of limited benefit compared to an actual source code review focusing on security flaws. These tasks are primarily provided as an alternative for those utilities and organizations that do not have access to the source code of the products they are testing. It is expected that very few utilities will perform this subcategory of penetration tasks. Figure 7 shows a typical task flow for analysing device firmware images in their binary format.

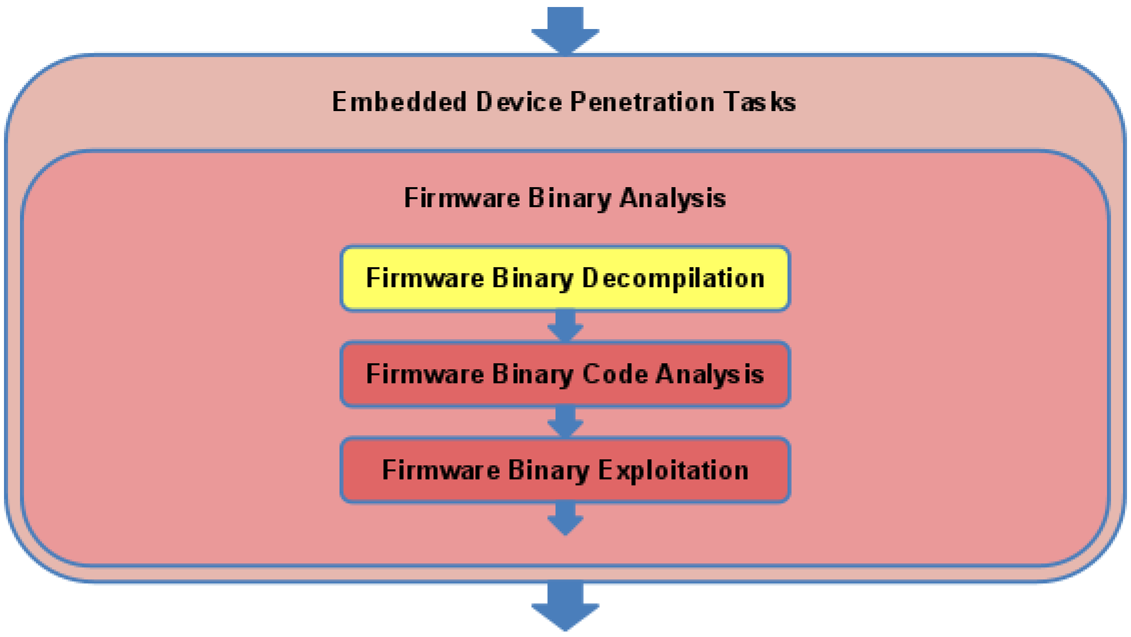

Embedded Device Penetration Tasks
Firmware Binary Analysis
Firmware Binary Decompilation
Firmware Binary Code Analysis
Firmware Binary Exploitation

Figure 7: Firmware Binary Analysis Task Flow

This subcategory of penetration tasks assumes the firmware was obtained in previous tasks or provided directly to the tester.

4.3.1 Firmware Binary Decompilation

Level of Effort: Medium

Task Description: If firmware is successfully retrieved and the tester has sufficient time and skill, decompile the firmware and attempt to identify vulnerabilities in the instruction calls. Warning, this task often proves very difficult as many microprocessors do not have publicly available decompilers. Consequently, one may need to be created first.

Task Goal: Obtain a human readable version of the firmware for later analysis.

4.3.2 Firmware Binary Code Analysis

Level of Effort: High to Extremely High

Task Description: Identify weaknesses in memory use, loop structures, cryptographic functions, interesting functions, etc.

Task Goal: Identify vulnerabilities that can be exploited.

4.3.3 Firmware Binary Exploitation

Level of Effort: High to Extremely High

Task Description: Based on the findings from previous steps, determine feasible attacks which can be launched at the firmware.

Task Goal: Create proof of concept attacks to demonstrate the feasibility and business risk created by the discovered vulnerabilities.

5 Network Communications Penetration Tasks

This section pertains to the testing of network communications for the smart grid systems, such as field area networks. Primary targets include wireless medium, network protocols, network segmentation controls, etc. The overarching goal is to identify vulnerabilities that allow an attacker to control network traffic or to subvert a device through protocol manipulation. In AMI systems, this communication can be meter-to-aggregator communication in the NAN, aggregator-to-headend in the WAN, meter-to-headend in direct communication architecture, and other communications between the headend and other systems such as the MDMS. These penetration tasks could also be used to test communication between the meter and the HAN.

Figure 8 below shows the overall process flow of the task sub-categories in this section. The figure shows the two task sub-categories may be performed in parallel. As in previous diagrams in this document, the colors represent the recommended likelihood that a utility should consider performing these task sub-categories, and the relative level of expertise required.

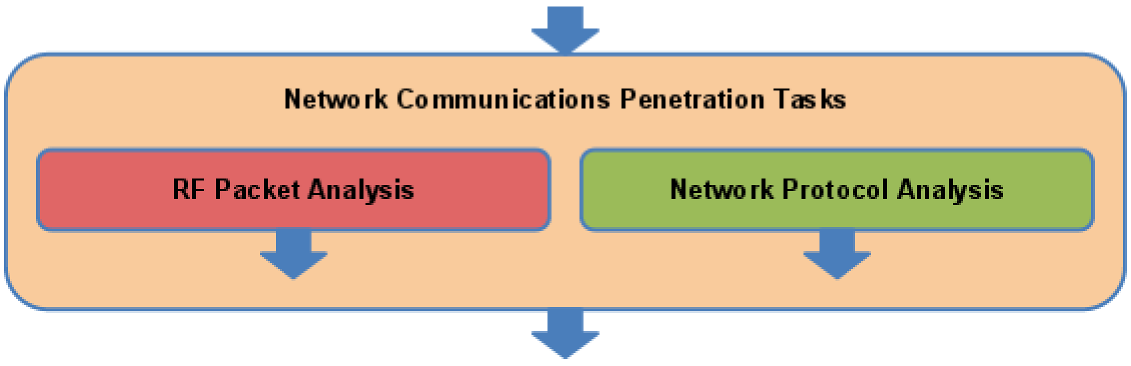

Network Communications Penetration Tasks
RF Packet Analysis
Network Protocol Analysis

Figure 8: Network Communications Subcategory Flow

Each subcategory below includes a similar diagram depicting the process flow and recommended likelihood to perform for each task.

<u>Suggested Tools</u>:

- Traffic capture and protocol decoder software such as Wireshark or tcpdump
- Hardware network taps
- Man-in-the-Middle tools such as Ettercap
- Protocol fuzzing tools such as Sulley
- Network packet generation such as Scapy
- Universal radio analysis kit, such as USRP2 with GNU Radio

5.1 RF Packet Analysis

This subcategory of penetration tasks focuses on the analysis of lower-layer RF communications such as frequency hopping, modulation, multiplexing, and data encoding in the Physical Layer and Medium Access Control Layer (PHY/MAC) of the Open Systems Interconnection (OSI) model. In an AMI system, this is usually the wireless communications between meters in the NAN. For AMI meters in the United States, this is usually proprietary to each vendor, often in the 900 MHz ISM band spectrum. These tests also pertain to cellular communications in the WAN.

It is usually assumed that network traffic can be extracted from captured RF communcations. Because of this, utilities often choose not to perform these sub-tasks and often skip to the next Network Protocol Analysis subcategory of tasks. However, some utilities may find this task subcategory useful to determine the level of effort it would take for an attacker to capture and decode their RF network traffic, especially when the utility knows of security weaknesses in the higher layer network protocols.

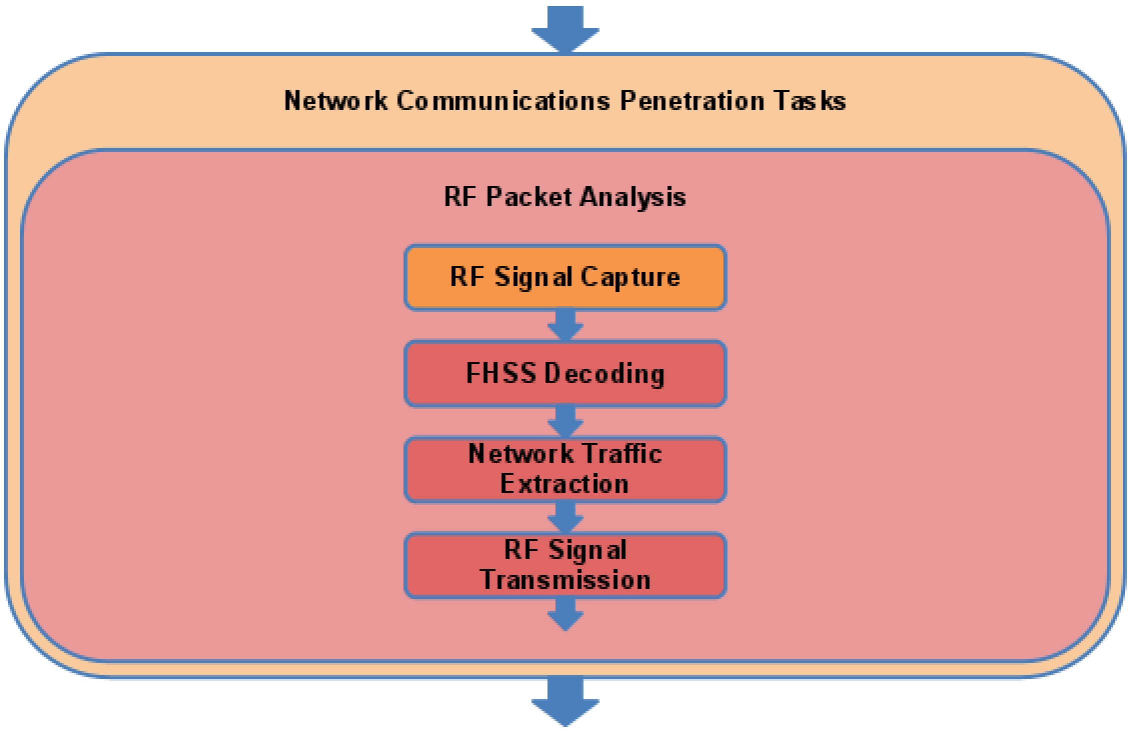


Figure 9: RF Packet Analysis Task Flow

5.1.1 RF Signal Capture

Level of Effort: Medium to High

Task Description: Use a tool (such as a USRP2) to capture the RF communications of the target field device.

Task Goal: Obtain data for following tasks.

5.1.2 FHSS Decoding

Level of Effort: Medium to High

Task Description: Use a tool to decode frequency-hopping pattern.

Task Goal: Obtain data for following tasks.

5.1.3 Network Traffic Extraction

Level of Effort: Medium to High

Task Description: Use a tool to decode and extract communications payload from RF capture.

Task Goal: Obtain data for following tasks.

5.1.4 RF Signal Transmission

Level of Effort: Medium to High

Task Description: Use a tool to transmit RF signals at the appropriate frequencies and hopping patterns to either replay captured data, impersonate the target field device, or attempting to cause denial of service scenarios.

Task Goal: Identify vulnerabilities in the RF signaling.

5.2 Network Protocol Analysis

This subcategory of penetration tasks focuses on analysis of network protocols above the PHY/MAC layer or from layer two and above in the OSI model. In AMI systems, this includes all communication between the headend and field devices and all communication between the headend and systems such as the MDMS or CIS.

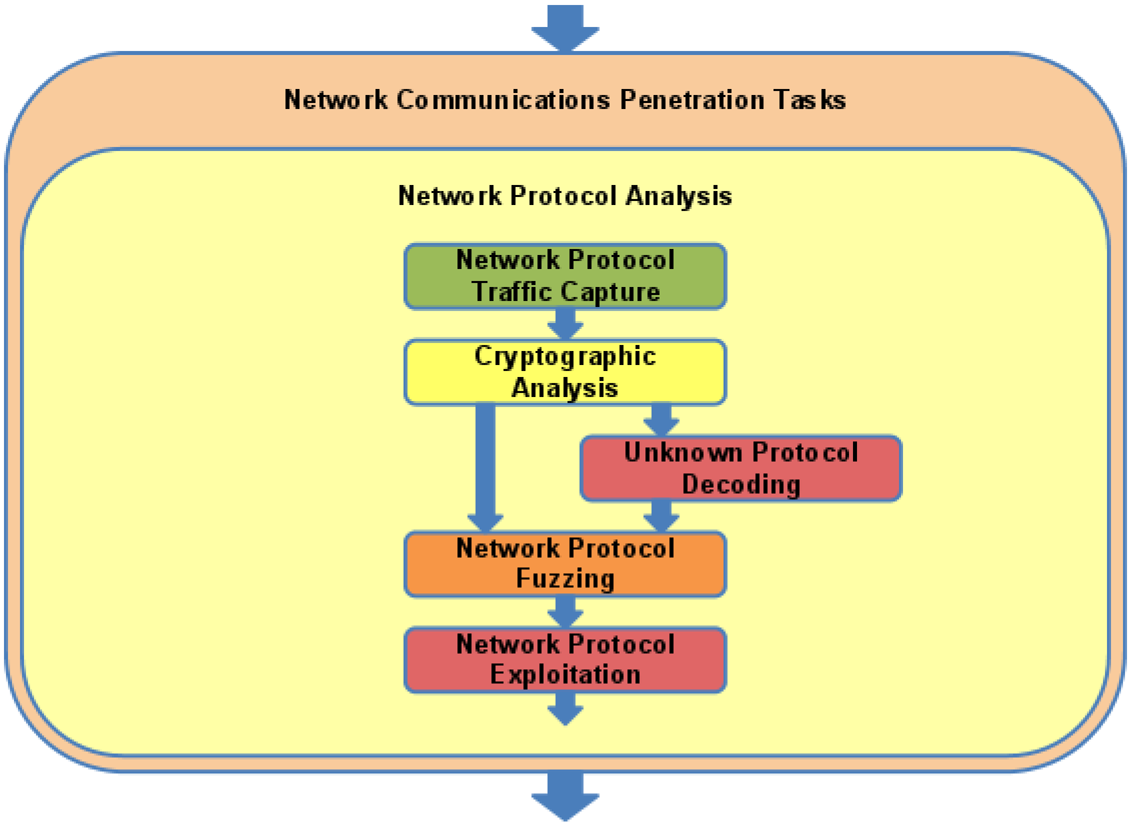

Network Communications Penetration Tasks
Network Protocol Analysis
Network Protocol Traffic Capture
Cryptographic Analysis
Unknown Protocol Decoding
Network Protocol Fuzzing
Network Protocol Exploitation

Figure 10: Network Protocol Analysis Task Flow

5.2.1 Network Protocol Traffic Capture

Level of Effort: Low

Task Description: Use a tool to capture sample communications. Attempt to cause known actions that result in communications between devices, such as firmware updates, and capture this communication individually to facilitate later analysis. Obtain samples of all target functionality.

Task Goal: Obtain data for the following tasks.

5.2.2 Network Protocol Cryptographic Analysis

Level of Effort: Low

Task Description: If the traffic capture uses a known protocol, identify the negotiated cryptographic algorithm and key length used to determine if any known vulnerabilities exist. If traffic capture is using an unknown protocol and is not readable, extract payloads from the captured network traffic and perform an entropy analysis to determine if the data is encrypted. High levels of entropy among the payload bytes often signify that encryption is being used, and weaknesses in cryptographic implementations can often be determined by variations in that entropy.

Task Goal: Determine if cryptographic is being used and identify any vulnerabilities.

5.2.3 Unknown Protocol Decoding

Level of Effort: High to Extremely High

Task Description: If traffic capture is using an unknown protocol, reverse engineer the network captures in an attempt to understand the protocol. Analyze each capture in light of the actions performed to initiate that traffic. For instance, if analyzing a traffic capture of a firmware update, try to identify the firmware being sent in the payload. Additionally, analyze actions such as when the meter joins a network to determine if an authentication mechanism is being used as well as the addressing scheme for the field area network.

Task Goal: Identify the purpose of blocks of data that could be used in later analysis.

5.2.4 Network Protocol Fuzzing

Level of Effort: Medium to High

Task Description: Use a tool to send both valid and invalid communications to both end points of the communications link individually, analyzing the results and identifying anomalies. This task includes items such as password guessing, invalid input testing, data enumeration, etc.

Task Goal: Identify vulnerabilities in the network protocol implementation.

5.2.5 Network Protocol Exploitation

Level of Effort: High to Extremely High

Task Description: Based on the findings from previous tasks, determine feasible attacks which can be launched on the field technician interface. For example, if devices are not required to authenticate themselves when joining a field area network, it may be possible to insert a 'rogue' node in the network or to harvest meters away from a legitimate data concentrator. Another example might be spoofing a firmware update or disconnect signal.

Task Goal: Create proof of concept attacks to demonstrate the feasibility and business risk created by the discovered vulnerabilities.

6 Server OS Penetration Tasks

This section pertains to the testing of the operating system of the control servers. This follows more traditional network-based vulnerability assessment of the windows, unix, and linux based systems, such as the identification of missing security patches, insecure configurations, or presence of insecure services. The overarching goal is to identify and exploit un-patched vulnerabilities to gain access to the control server.

Figure 11 below shows the overall process flow of the task sub-categories in this section. The figure shows the three task sub-categories must be performed in series. As in previous diagrams in this document, the colors represent the recommended likelihood that a utility should consider performing these task sub-categories, and the relative level of expertise required.

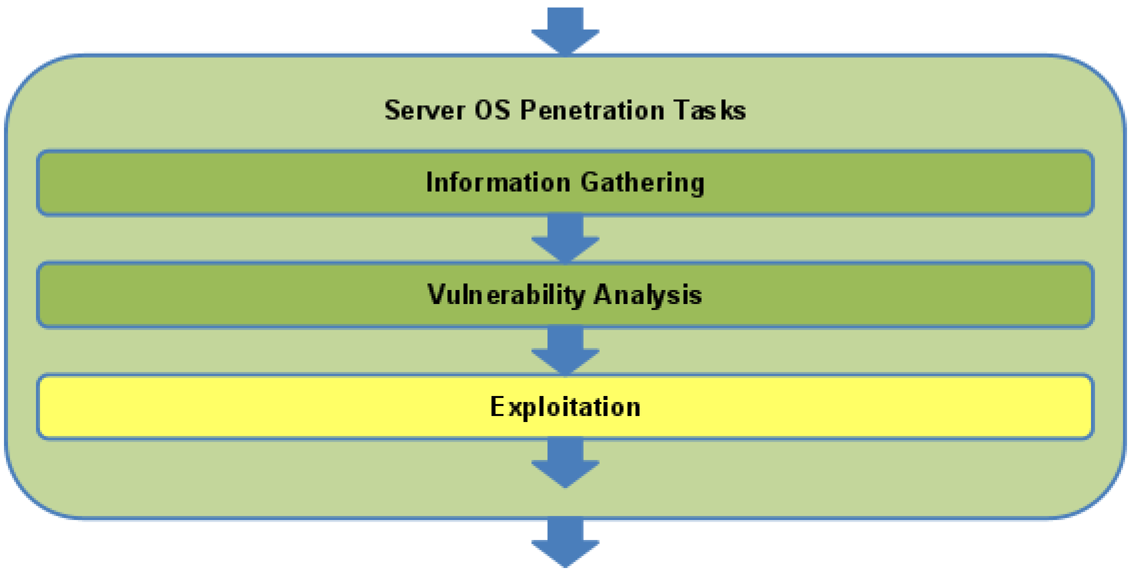


Figure 11: Server OS Subcategory Flow

Each subcategory below will include a similar diagram depicting the process flow and recommended likelihood to perform for each task.

<u>Suggested Tools</u>:

- Standard network vulnerability assessment and penetration testing tools such as found on the Backtrack distribution

6.1 Information Gathering

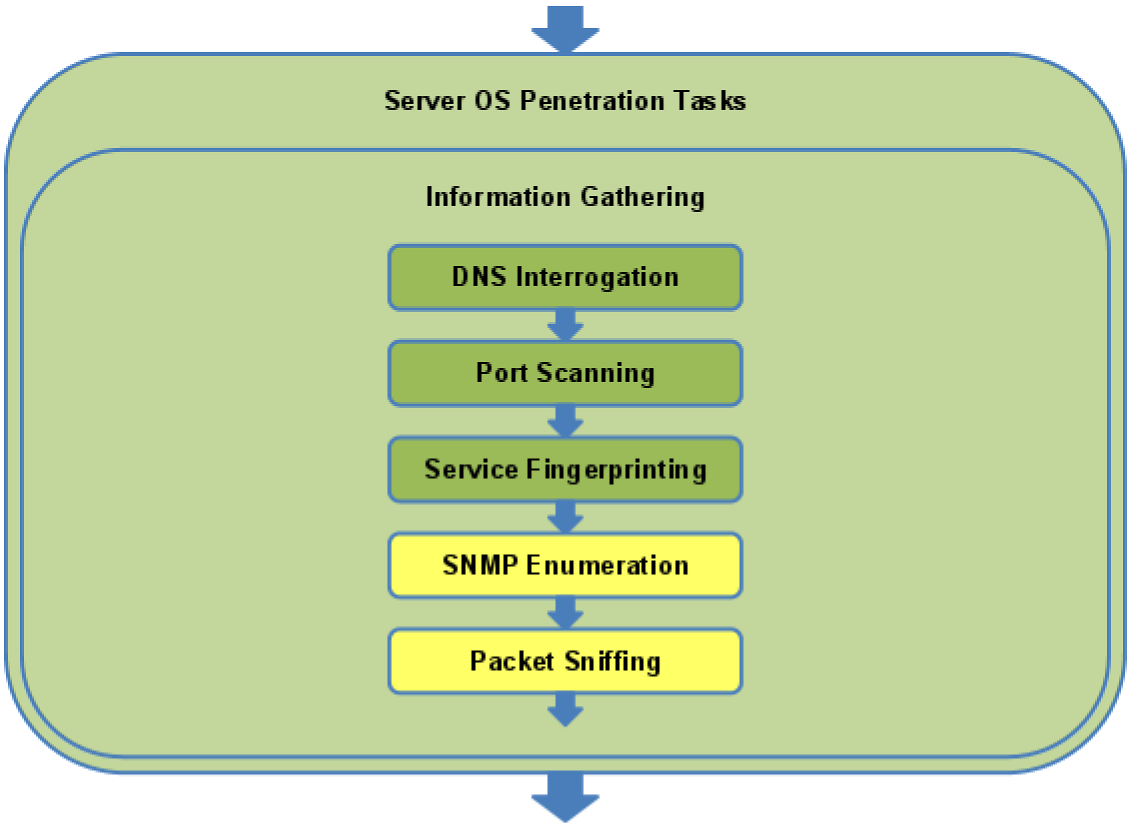


Figure 12: OS Information Gathering Task Flow

6.1.1 DNS Interrogation

Level of Effort: Low

Task Description: Use tools to attempt zone transfers and perform queries from target Domain Name Service (DNS) servers.

Task Goal: Identify targets, verify ownership, and detect anomalies.

6.1.2 Port Scanning

Level of Effort: Low

Task Description: Use tools that send requests to possible application layer services (such as scanning TCP and UDP ports to discover services like HTTP and SSH).

Task Goal: Identify all listening services and possible firewall rules.

6.1.3 Service Fingerprinting

Level of Effort: Low

Task Description: Use tools to examine listening services.

Task Goal: Identify the nature and function of all listening services.

6.1.4 SNMP Enumeration

Level of Effort: Low

Task Description: Use tools to attempt to examine SNMP services.

Task Goal: Identify insecure SNMP services, extract information about the endpoints, and identify vulnerabilities that allow attackers to reconfigure endpoints.

6.1.5 Packet Sniffing

Level of Effort: Low

Task Description: Capture various samples of network communications.

Task Goal: Collect samples for later analysis.

6.2 Vulnerability Analysis

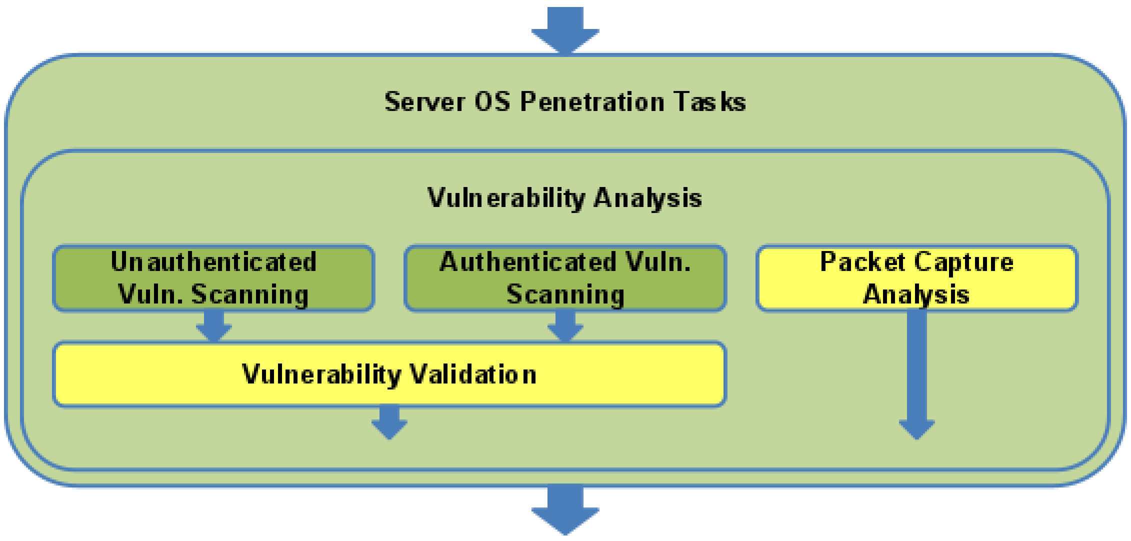

Server OS Penetration Tasks
Vulnerability Analysis
Unauthenticated Vuln. Scanning
Authenticated Vuln. Scanning
Packet Capture Analysis
Vulnerability Validation

Figure 13: OS Vulnerability Analysis Task Flow

6.2.1 Unauthenticated Vulnerability Scanning

Level of Effort: Medium

Task Description: Use automated tools without credentials to identify known vulnerabilities in network services and their respective systems.

Task Goal: Identify vulnerabilities in the operating system and the network services

6.2.2 Authenticated Vulnerability Scanning

Level of Effort: Medium

Task Description: Use automated tools that use valid credentials to authenticate to systems and identify known vulnerabilities with installed software.

Task Goal: Identify vulnerabilities in the operating system and installed software.

6.2.3 Vulnerability Validation

Level of Effort: Medium

Task Description: Manually validate findings from automated tools where possible. Merge and combine findings where applicable.

Task Goal: Consolidate findings and remove any false positive findings that you identify.

6.2.4 Packet Capture Analysis

Level of Effort: Low to Medium

Task Description: Examine network traffic samples and look for protocols with known vulnerabilities such as session hijacking, weak authentication, or weak/no cryptographic protections.

Task Goal: Identify vulnerabilities in network protocols and network communications.

6.3 Server OS Exploitation

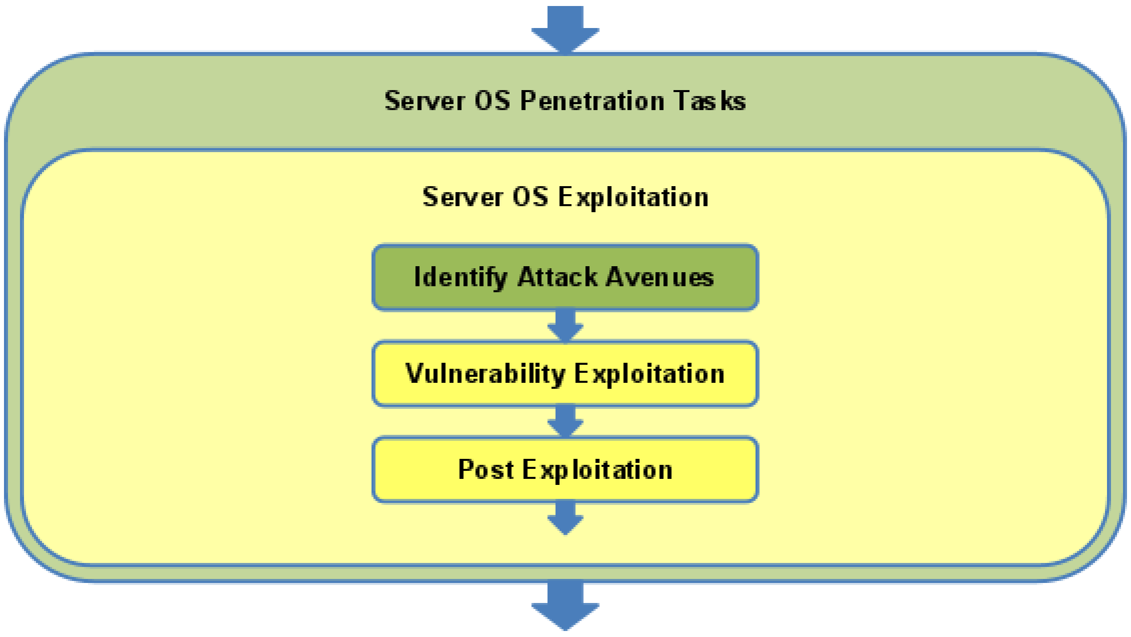


Figure 14: Server OS Exploitation Task Flow

6.3.1 Identify Attack Avenues

Level of Effort: Medium

Task Description: Review all findings and outputs from previous tasks and identify plausible attacks that have a moderate chance of success. Prioritize these possible attacks by likelihood and the tester’s ability to execute them.

Task Goal: Organize and plan next steps.

6.3.2 Vulnerability Exploitation

Level of Effort: Low to Medium

Task Description: Create proof of concept attacks to demonstrate the feasibility and business risk created by the discovered vulnerabilities. Once a vulnerability has been exploited, attempt to pivot and identify additional vulnerabilities to exploit.

Task Goal: Validate the assumed business risk created by the identified vulnerabilities and identify additional targets of opportunity.

6.3.3 Post Exploitation

Level of Effort: Low to Medium

Task Description: Remove any code, data, or configurations that were added to the system as a part of the assessment.

Task Goal: Return the systems to their pre-assessment state.

7 Server Application Penetration Tasks

This section pertains to the testing of applications that are executing on the control server. Standard software testing guidelines such as the Open Web Application Security Project (OWASP) Testing Guide can be leveraged to perform this task. The overarching goal is to identify vulnerabilities in applications that allow an attacker to gain access to the control server. In AMI systems, these applications will be web-based interfaces and web services hosted by the headend (or one of the sometimes numerous servers that makes up the "headend"). These penetration tasks can also be extended to the MDMS and other "control servers" that interact with the headend.

Figure 15 below shows the overall process flow of the task sub-categories in this section. The figure shows the three task sub-categories must be performed in series. As in previous diagrams in this document, the colors represent the recommended likelihood that a utility should consider performing these task sub-categories, and the relative level of expertise required.

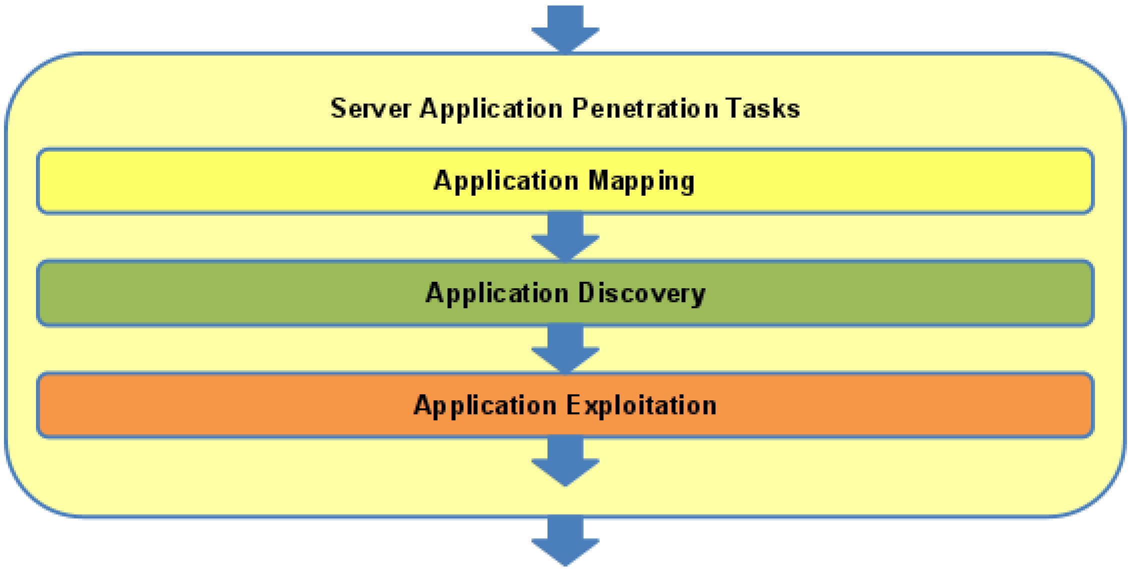


Figure 15: Server Application Subcategory Flow

Each subcategory below will include a similar diagram depicting the process flow and recommended likelihood to perform for each task.

<u>Suggested Tools</u>:

- Web application penetration testing software such as found on the Samurai Web Testing Framework (SamuraiWTF) project

7.1 Application Mapping

This subcategory of penetration tasks focuses on the gathering of information and allows the tester to gain a firm understanding of the user interface or web service functionality and design.

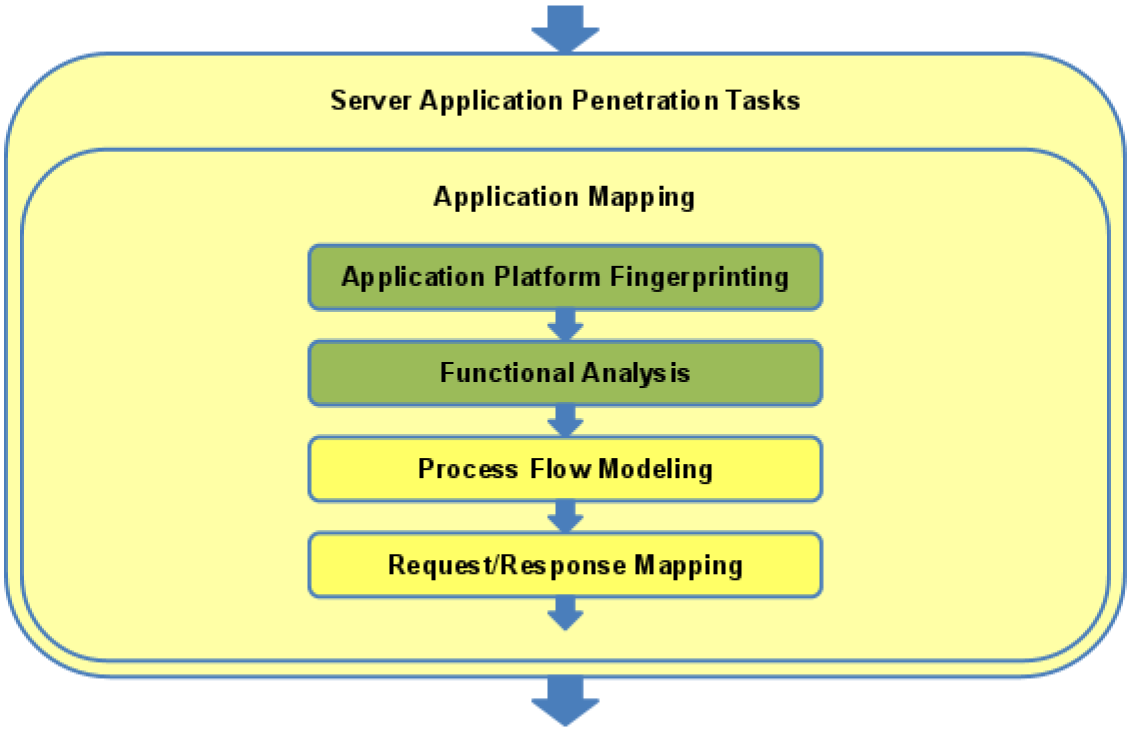


Figure 16: Application Mapping Task Flow

7.1.1 Application Platform Fingerprinting

Level of Effort: Low

Task Description: Use tools to query the application service to identify the platform type and version hosting the application. (Such as Apache and Tomcat)

Task Goal: Identify the application server and technologies used to host the application.

7.1.2 Functional Analysis

Level of Effort: Low

Task Description: Gain an understanding of the application from the user's perspective. Explore the application and identify major functionality and features exposed to the user. Identify major sections and portions of the application, including the user roles.

Task Goal: Gain a better understanding of the application for later analysis.

7.1.3 Process Flow Modeling

Level of Effort: Low

Task Description: Model the process flows that users must follow while using the application. Identify dependencies between actions and requirements to get to each portion of the application.

Task Goal: Gain a better understanding of the application for later analysis.

7.1.4 Request/Resource Mapping

Level of Effort: Low

Task Description: Attempt to map, execute, and record every possible request in the application. Examine the requests and responses to understand how the application works from the developer's perspective. Identify parameter names and values that are reflected back to the user or appear to be used in a database query.

Task Goal: Identify requests that have a higher probability of containing vulnerabilities. Prioritize for later analysis.

7.2 Application Discovery

This subcategory of penetration tasks focuses on the identification of vulnerabilities in the user interfaces or web services.

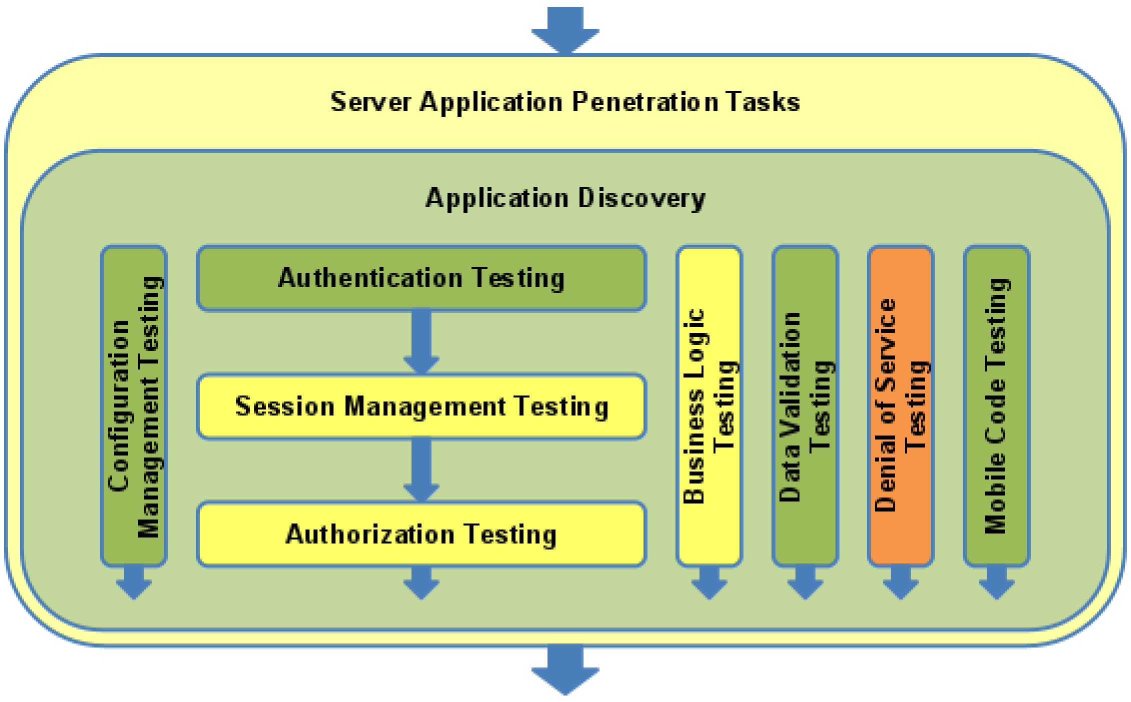


Figure 17: Application Discovery Task Flow

7.2.1 Configuration Management Testing

Level of Effort: Low

Task Description: Test the platform and application server configuration, such as SSL/TLS testing, file extension handling, method handling, and the existence of administrative interface and unreferenced links.

Task Goal: Identify vulnerabilities in the application.

7.2.2 Authentication Testing

Level of Effort: Low

Task Description: Test the application authentication for flaws such as user enumeration, guessable passwords, authentication bypass, flawed password reset, race conditions, multifactor authentication, and CAPTCHA implementation weaknesses.

Task Goal: Identify vulnerabilities in the application.

7.2.3 Session Management Testing

Level of Effort: Low

Task Description: Test the application for session management flaws such as session fixation, session hijacking, unprotected session keys, and Cross Site Request Forgery (CSRF).

Task Goal: Identify vulnerabilities in the application.

7.2.4 Authorization Testing

Level of Effort: Low

Task Description: Test the application for authorization flaws such as path traversal, authorization bypass, and privilege escalation.

Task Goal: Identify vulnerabilities in the application.

7.2.5 Business Logic Testing

Level of Effort: Low

Task Description: Test the business logic flow and user process flow to verify steps that cannot be skipped or re-ordered.

Task Goal: Identify vulnerabilities in the application.

7.2.6 Data Validation Testing

Level of Effort: Low

Task Description: Test the application for data validation flaws such as XSS, SQL Injection, LDAP injection, XPath Injection, overflows, format string issues, and HTTP Splitting.

Task Goal: Identify vulnerabilities in the application.

7.2.7 Denial of Service Testing

Level of Effort: Low

Task Description: Test the application for flaws that may cause denial of service vulnerabilities either on the service platform, in the application logic, or on the backend systems and databases.

Task Goal: Identify vulnerabilities in the application.

7.2.8 Mobile Code Testing

Level of Effort: Low

Task Description: Test the application for flaws in the use of mobile or client-side code.

Task Goal: Identify vulnerabilities in the application.

7.3 Application Exploitation

This subset of penetration tasks focuses on the exploitation of vulnerabilities found in the previous tasks and the escalation of access the tester has in the application.

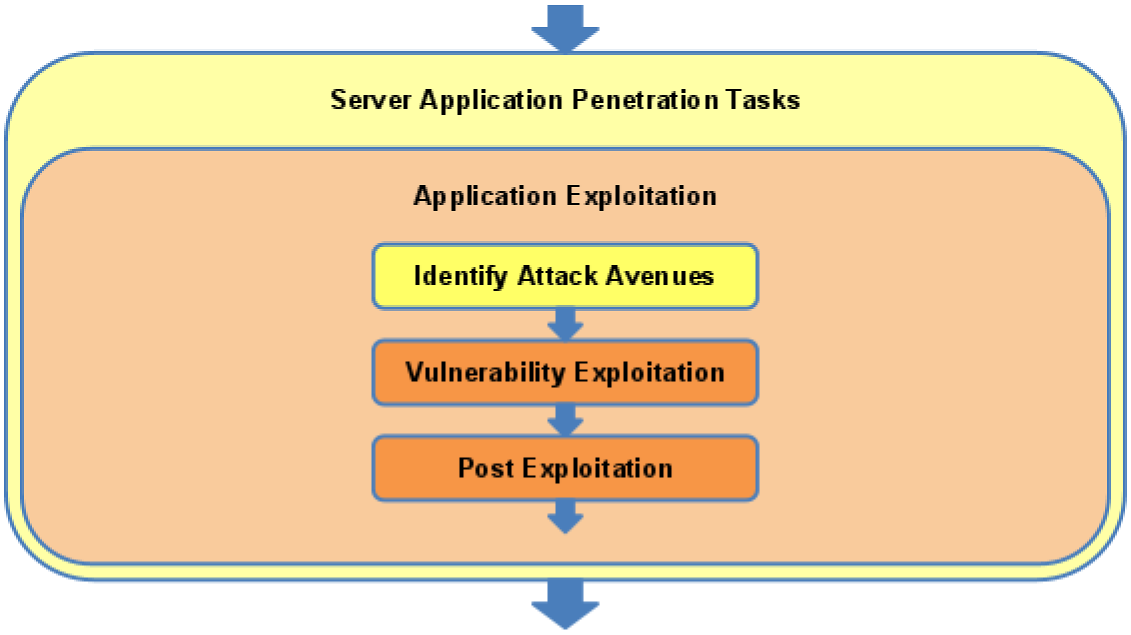


Figure 18: Application Exploitation Task Flow

7.3.1 Identify Attack Avenues

Level of Effort: Medium

Task Description: Review all findings and outputs from previous tasks and identify plausible attacks that have a moderate chance of success. Prioritize these possible attacks by likelihood and the tester's ability to execute them.

Task Goal: Organize and plan next steps.

7.3.2 Vulnerability Exploitation

Level of Effort: Low to Medium

Task Description: Create proof of concept attacks to demonstrate the feasibility and business risk created by the discovered vulnerabilities. Once a vulnerability has been exploited, attempt to pivot and identify additional vulnerabilities to exploit.

Task Goal: Validate the assumed business risks created by the identified vulnerabilities and identify additional targets of opportunity.

7.3.3 Post Exploitation

Level of Effort: Low to Medium

Task Description: Remove any code, date, or configurations that were added to the system as a part of the assessment.

Task Goal: Return systems to their pre-assessment state.

8 End-to-End Penetration Test Analysis

The final task in any penetration test should be a gap analysis of communications that span the entire system. This should include a review of input and output from external systems that may not be in scope for this assessment. For instance, when testing an AMI meter system, a tester might have performed tests on all components from the meter to the headend. However this final end-to-end task should ensure that all possible inputs from external systems to in-scope systems have been tested and evaluated as possible attack angles, such as an out-of-scope MDMS or CIS system connecting with an in-scope headend. Penetration testers should also identity if any vulnerabilities found later in the testing process affect components tested earlier or by other other testing teams.

9 Result Interpretation and Reporting

As penetration-testing tasks are completed, vulnerabilities should be found and documented. When a vulnerability is found, testers should briefly document the relative risk that the particular vulnerability presents to the in-scope system and the business in general and a brief note of how that vulnerability could be mitigated. These initial impressions of risk and mitigation are important to document at that time since the tester is usually most immersed in that vulnerability at the time when it is discovered. Upon completion of all penetration test tasks, theses initial impressions should be reviewed and adjusted based on other vulnerabilities found in the system. For instance, a penetration tester may find two vulnerabilities that he initially believes are a low risk vulnerabilities to the system. However upon completion of all penetration testing tasks, the tester may realize that an attacker could leverage both low level vulnerabilities to create a new high risk vulnerability. This analysis should be done once all testing tasks are completed and the final report is being generated. At the time of final report generation, each vulnerability should be more fully documented and mitigation recommendations should be expanded.

The final report should, at a minimum, include the following sections:

- Executive Summary - a brief 1-2 page section discussing the overarching root causes for the vulnerabilities and high level business strategies to address these root causes.
- Introduction – a short section describing the goals of the tests, components that were in and out of scope, any special restrictions on the tests, and the team involved with the testing.
- Methodology – a short section of the report focuses on the technical reasons for the test as well as the methodology used.
- Findings and Recommendations – this section of the report is traditionally the longest, most detailed, and highly technical. This is the core of the report for future use and reference. This section may also discuss the likelihood and impact of each vulnerability within the context of the proposed or existing deployment.
- Conclusion – a section similar to the executive summary but at a more technical depth summarizing the major findings and recommendations. This section should also discuss any major questions or goals of the assessment such as the team's recommendations of a go no-go purchase of a product.