This document contains only the question and my answers from the challenge. For more details see the end report.

Content

1. Remarks

The numbers reference to the respective attack step / observation in section three in the end report.

2. Solution

2.1. Unsuccessful Attacks

Question: "Please describe each of the unsuccessful attacks Mr. Potter's goons attempted against Bedford Falls infrastructure."

Answer:

- #7, #4: spear phishing email sent to Don; email is received but embedded XSS didn't work
- #13: attacker 10.21.22.253 accessed traffic light 10.21.22.23 on controller port, presumably using modbus, seemed to have failed
- #15: not all data base information could be altered by attacker in the web application
- #16: ARP spoofing for 10.25.22.22 and 10.25.22.30 didn't work
- #19: attacker 10.25.22.252 accessed web server on 10.25.22.30, but didn't seem to find anything interesting
- #20: attacker 10.25.22.252 accessed web server on 10.25.22.22, basic information could be retrieved, but more sensitive information was basically protected (Digest Authentication); default credentials did not work

2.2. Defenses in Place

Question: "What defenses did George deploy that thwarted those attacks?"

Answer:

- #7, #4: potentially: XSS filtering or hardened / patched web application
- #13: potentially default credentials had been changed / access locked down
- #15: web application data base user didn't have full rights for all tables (only data necessary for the operation of the web app could be changed)
- #16: potentially static MAC entries for critical systems / intelligent network equipment preventing MAC spoofing
- #19: web server not used
- #20: authentication had been enabled and default credentials had been changed

2.3. Damage

Question: "How had Mr. Potter caused the power grid outage that made George consider jumping off of the bridge?"

Answer:

- attacker sent a spear phishing email to Don which caused Don downloading and executing (on train management workstation, 10.25.22.253) a malicious file thinking of it being a patch for a security vulnerability
- the malicious file opened a connection from the train management workstation (10.25.22.253) to attacker's system (10.21.22.253)
- attacker could now use the train management workstation as a stepping stone, doing reconnaissance

- reconnaissance found weak Windows password for administrative user "ernie" on windows workstation 10.25.22.58
- this account was used by the attacker to remotely upload a file (VNC server) and install and run it as a service on the workstation (10.25.22.58)
- the attacker now could access this workstation (10.25.22.58) via the train management workstation (10.25.22.253) from his system (10.21.22.253)
- the workstation (10.25.22.58) happened to be unlocked and having a monitoring / control application running which remotely controlled 10.25.22.20
- so the attacker could remotely operate the tool on the workstation 10.25.22.58 and shut down the generator power using the control workstation 10.25.22.20 (possibly re-using credentials for "ernie")

2.4. Suggested Defenses

Question: "What defenses could George have employed to prevent Mr. Potter's power grid attack?"

Answer:

The following defenses would have prevented the attack on the power grid:

- conduct security awareness training for employees (e.g. to spot phishing emails)
- deploy patching procedure / workflow to prevent installation of unknown files
- only allow necessary connections between dedicated systems and ports across network boundaries for critical segments
- harden Windows installation; don't allow weak passwords (e.g. for user "ernie")
- implement account lockout / brute force detection on Windows systems / workstations
- don't allow administrative privileges when not absolutely necessary (e.g. for user "ernie")
- implement anti-virus solution for basic protection where possible; otherwise move systems into separate protected network and allow access for valid systems only
- use application whitelisting for critical systems to allow only execution of well-known software
- don't allow systems being unlocked when not interactively used (esp. control workstations)
- require re-authentication after a reasonable time out for critical operations
- require a two factor authentication / approval workflow for critical operations
- avoid reusing credentials; better use dedicated user IDs for different role if feasible

Furthermore the following measures are suggested (for details see end report):

- implement encryption (https) and authentication / authorization (logins, user / role management) also for internal applications
- prohibit browsing from working systems to non-internal destinations
- implement dedicated and isolated (virtual) surfing infrastructure for access to the Internet through a proxy
- DNS should only be allowed to internal DNS server / proxy which act as an application level gateway and allows filtering etc.
- implement internal update and patching / software distribution infrastructure
- disallow uncontrolled software installation / updating for staff

- restrict software installation on working systems
- encrypt email traffic (authentication and delivery)
- tighten physical security to network resources and add logical protection (e.g. NAC) so that unknown systems can't be plugged into the network
- use fixed MAC resolution on critical systems
- set up network monitoring to detect duplicate usage of IP and MAC addresses
- critical systems should not be able to connect to systems outside the network segment (except to limited control stations in another internal network)
- only allow dedicated systems to connect to sensitive infrastructure (e.g. set up IP based ACLs)
- web application waterqual.publicworks.city.nw ("CyberCity Water Monitoring & Alarm System"):
    - harden web app against session riding attacks; disallow "remember me" functions
    - only allow encrypted web traffic (https)
    - pentest internal applications; implement secure development lifecycle (SDL); implement best practices
    - harden data base / web server installation so that data base can't write files (and especially not into web server's Document Root)
    - harden web server installation so that the web server can't create files in the Document Root
    - harden PHP installation / use additional tools to remediate weaknesses of PHP and prohibit insecure functions to be executed
    - place data base credentials outside Document Root folders
- protect web interfaces on networked systems by encryption and authentication / authorization if possible; otherwise move them into separate protected network and allow access for valid systems only
- implement brute force protection for account validations
- rework network monitoring infrastructure to make sure no frames are being missed for captures
- set up host monitoring (HIDS / HIPS) to detect new files / services / system changes / unusual log entries
- set up network (NIDS / NIPS) and host monitoring (HIDS / HIPS) to detect port scans
- set up honeypots to find and analyze malicious activities