Cayleigh Goberman

6/2/2025

Goal-Oriented Response Planning

Notes

**Timesheet Notes**

|  |  |
| --- | --- |
| **Date** | **Hours** |
| 6/2/2025 | 9:00 am – 12:00 pm |
|  | 12:30 pm – 5:30 pm |

**Notes**

* Use an adapted goal-oriented action planning program to address problems
* Have multiple threat scenarios to simulate (Scenario A, Scenario B, etc)
* Focus on things in the sent email
* Possible Focuses:
  + Authentication? --> Not included in original email
    - Something you know
      * Password or phrase
      * PIN
    - Something you have
      * Code from app or SMS
      * Push notification
      * USB token
    - Something you are
      * Finger or thumb print
      * Face scan
      * Iris scan
    - Just one of the three types is usually insecure, but two of the three is considered secure (two-factor authentication)
    - **Implementation Ideas**
      * Fairly simple. To begin with, we can use a password to access the agent. If we want to be more secure, we could add two-factor authentication.
  + Scan and close any ports
    - Common practice to close unused ports
    - **Implementation Ideas**
      * Send packages to different ports and analyze their responses to determine if the port is open or closed
      * Close ports that are not in use
        + Filter open ports? Most of the time, data from non-recognized addresses should not be allowed in?
  + Check for sudo in scripts/commands
    - sudo: Allows a normal user to run a program with the security privileges of another user
    - **Implementation Ideas**
      * Check if the user is a sudo user or an administrator
      * Require all sudo users to enter a password. If the user changes sudo configurations, reset the configurations to default after the sudo user exits the application.
  + Reverse shells
    - A shell that is running on one computer but accepts requests and relays responses to another computer
    - A type of cyber-attack in which the targeted system is tricked into establishing a connection to the attacker’s computer
    - Allows attackers to remotely control the system
    - **Implementation Ideas**
      * Check for strange outbound traffic
      * Check connection attempts. Report to administrator if they come from a suspicious location (Ex: A Marist computer is trying to connect to a computer located outside of Marist University)
      * Monitor general traffic. Report to administrator if there is an unusual uptick in traffic. Maybe have the agent close the affected ports for a little bit?
  + Check file integrity
    - Checks a file’s credentials, privileges and security settings, content, core attributes and size, hash values, and configuration variables
    - Implementation varies (I think), but we just have to check the core components of a file. Maybe delete it if it has an unauthorized change
  + Check ARP table for odd entries (multiple IPs mapping to one MAC)
    - **Implementation Ideas**
      * Keep a list of all IP addresses and periodically run through it. If a single device is listed under multiple IP addresses, temporarily block those IP addresses and report to network administrator.
  + Domain Name System (DNS) high jacking
    - DNS translates domain names into IP addresses
      * Ex: “nytimes.com” is translated into an IP address when typed into a browser so users do not have to remember the IP address
    - DNS Tunneling: Uses DNS protocol to communicate non-DNS traffic over the default port for DNS traffic, port 53.
      * Normally sends HTTP and other protocol traffic
      * Can be used to disguise outbound traffic as DNS, concealing data that is typically shared through an internet connection
      * Extract data from a system into the attacker’s infrastructure
      * Can also be used for command and control callbacks from the attacker’s infrastructure into the system
    - DNS Poisoning: Fake information is entered into the cache of a domain name server, resulting in DNS queries producing an incorrect reply and sending users to the wrong website
    - **Implementation Ideas**
      * Hide primary DNS server from public access via network isolation and firewalls
      * Check the information contained in DNS messages. If it doesn’t look like it should be there, investigate.
        + Don’t process input if it isn’t checked as being:

SQL/JavaScript/Any injection attack

Within the correct length

Within the correct data type

* + - * + Close used port, isolate/block associated IP address, report to network administration.
        + If we want to get really in-depth, run a virus scan or something similar.
      * Filter DNS requests and block malicious domains
      * Only allow network administrators to access the DNS?
      * If something fishy is detected, close the outgoing port while the agent isolates the issue?
  + Check for outdated packages
    - A package is outdated if it no longer receives updates or support from developers
      * Can have known security vulnerabilities that new, updated packages have fixed
    - **Implementation Ideas**
      * Routinely check packages. If a package is not on a list of supported packages, delete it.
      * If a package stops being supported, remove it from the list of supported packages and run through the packages to delete them.
  + Check for hidden files and directories
    - **Implementation Ideas**
      * Regularly run a virus scan on the hidden files
      * Check to see if their size has increased suspiciously since they were last checked?
  + Check for gaps in logs
    - **Implementation Ideas**
      * Have the agent log everything it does
      * Have a clear amount of time logs must be retained
      * Occasionally review logs
      * Analyze the logs for gaps. If a gap is found, find the file associated with the gap and run a virus scan to check for suspicious activity.
* Extra Notes:
  + Be very careful about preventing memory leaks
    - Some malicious programs hoard memory. For example, a DDoS attack is just a form of resource exhaustion.
  + Pointers: An area of memory that points to another area of memory
    - Do not let users/programs...
      * Change where the pointer is pointing to
      * Receive a null value back
      * Force the program to print out debug info
      * Bypass some security feature
  + Use locks to prevent other threads from accessing a variable or value while this thread is using it
* Research how to put together requirements document
  + Essentials
  + Should have
  + Would be nice to have, but not very important for this project (Ex: Authentication)