Results of GG sir lab Test

Real Time Testing of IDS algorithms on GG sir lab

Date:- 13 June 2024

Attack Overview:

This attack utilized Python scripting with libraries such as Scapy, Pyshark, and pyasn1 to manipulate GOOSE protocol packets. The attack was conducted on a real time environment to assess the protocol's susceptibility to malicious packet injections.

Attack Script Operation

1. **Packet Capture**:
   - The script captured GOOSE packets from a live network interface then it reads them from a `pcapfile`.
2. **Packet Filtering**:
   - It filtered out GOOSE packets based on the `EtherType` (0x88b8) and of length 400 to 600.
3. **Packet Decoding**:
   - The captured GOOSE packets are decoded to extract important fields like the dataset, sequence number (`sqNum`), state number (`stNum`), and timestamp.
4. **Packet Manipulation**:
   - Users select specific packets to modify.
   - The script alters the `sqNum` and `stNum` fields, which are crucial for ensuring message sequence and state integrity.
     - Set `stNum` to a value calculated as `int(tmpSTNUM) + random.randint(0, 10000)`, and raise `sqNum` by 1000 compared to its previous value.

- It also changed the `allData` payload values to toggle from the previous ones.

  (e.g., setting `False` to `True`).

- The `timestamp` is also updated to the current time to make the packets appear recent.

5. **Sending Malicious Packets**:

   - The modified packets are sent out repeatedly at a high frequency (`1000` times) to the target device.

Prerequisites

> ⚠ First, ensure you have the attack script and the `goose` directory in the same directory. This setup is required for the script to locate and import the necessary modules from the `goose` directory.

Running the Script

To execute the script for the given `pcapfile`, use the following command in your terminal:

```
python3 script_name.py --pcapfile example.pcap --output output.pcap
```

Replace `script_name.py` with the actual name of your script, `example.pcap` with your input pcap file, and `output.pcap` with your desired output file name.

For live capture and attacking, use the following commands in your terminal:

```
touch output.pcapng
```

```
python3 script_name.py --livecapture --output output.pcapng
```

Replace `script_name.py` with the actual name of your script

Script Execution Breakdown

1. **Argument Parsing**:
   - If no arguments or invalid arguments are provided, the script will print:

     ```
     [-] For help menu, use -h
     ```

2. **Reading a PCAP File**:
   - If the `--pcapfile` argument is provided, the script reads the specified pcap file:

     ```
     [+] Reading pcap file...
     ```

3. **Live Capture**:
   - If the `--livecapture` argument is provided, the script starts a live packet capture:

     ```
     [+] Listening on eth1
     [+] YOU SHOULD HAVE SUDO PERMISSIONS TO DO THIS
     How many packets do you want to capture?: [user inpu
     t]
     [+] Capturing packets...
     [+] Capturing done..
     [+] Using the pcap file just captured..
     ```

4. **Parsing GOOSE Headers**:

```
[+] Parsing GOOSE header and GOOSE PDU header
[+] Decoding the GOOSE PDU header
```

5. **Printing a Sample Packet**:
   - Prompt to print a sample packet:

     ```
     Do you want to print a sample packet?[Y/N]:   [user i
     nput]
     ```

   - If the user inputs 'Y':

     ```
     [+] Printing the packet details...
     ```

6. **Scanning Unique MAC Addresses**:

   ```
   [+] Scanning for unique source-destination connection
   s...
   Unique source and destination MAC addresses:
   Source: [source MAC] --> Destination: [destination MAC]
   ```

7. **Selecting a Target Destination**:

   ```
   Select the destination you want to target in the range
   [1 to n]:   [user input]
   ```

8. **Filtering Packets**:
   - After selecting the target destination:

     ```
     [+] Selected source MAC address:   [source MAC]
     [+] Selected destination MAC address:   [destination
     MAC]
     ```

9. **Selecting Packets to Manipulate**:

```
How many packets do you want to manipulate:    [user inp
ut]
[+] Select the packets you want to manipulate in the ran
ge of 0 to [total packets].
Enter multiple packet numbers separated by space:    [us
er input]
```

10. **Printing Modified Data**:

- Prompt to print the modified data:

```
Do you want to print the modified data?[Y/N]?: [user
input]
```

Example:

Original Data

```
gocbRef: simpleIOGenericI
timeAllowedtoLive: 3000
datSet: simpleIOGenericIO
goID: events
t: 2023-02-20 15:26:58
stNum: 1
sqNum: 8
test: False
confRev: 2
ndsCom: False
numDatSetEntries: 8
allData
    False
    0000000000000
    False
    0000000000000
    False
    0000000000000
    False
    0000000000000
```

Modified Data

```
gocbRef: simpleIOGenericI
timeAllowedtoLive: 3000
datSet: simpleIOGenericIO
goID: events
t: 2024-06-11 18:34:52
stNum: 8011
sqNum: 508
test: False
confRev: 2
ndsCom: False
numDatSetEntries: 8
allData
    True
    0000000000000
    True
    0000000000000
    True
    0000000000000
    True
    0000000000000
```

11. **Sending or Saving Modified Packets**:
    - If the user chooses to send the modified packets:

      ```
      Do you want to send the modified packets? (time will be changed)[Y/N]: [user ip]
      ```

    - If sending is chosen:

      ```
      [+] Sending malicious packets...
      ```

    - If saving is chosen:

      ```
      Do you want to save the modified packets to a file? [Y/N]:   [user input]
      Set filename:   [user input]

      [+] Saving the modified packets in pcapng file..
      [+] Successfully saved pcapng file as [filename].pcapng !
      ```

12. **Exiting**:
    - If the script is interrupted:

      ```
      [-] Exiting
      ```

    - If stopping the injection:

      ```
      [-] Stopping the injection...
      ```

Impact of this Attack:

During testing on the live testbed, the attack script successfully manipulated GOOSE protocol packets by altering critical parameters (`sqNum` and `stNum`). This manipulation aimed to disrupt data synchronization and control logic among IEDs.

Despite attempts to obfuscate these modifications, the Intrusion Detection System (IDS) deployed within the network environment detected the attack.

Conclusion

IDS algorithm Successfully Detected the Attack.

Appendix

- Code snippet: `Attack_code_high_stNum_high_sqNum.py`
- Packet capture sample: `goose_real_time_capture.pcapng`