Report: Bluetooth Device Encryption Verification

Introduction

This report describes the process of verifying Bluetooth device encryption by capturing and analyzing Bluetooth packets. The focus is on identifying encryption-related commands and events using `pyshark` for packet analysis.

Checking for Encryption

The core functionality for verifying encryption in Bluetooth communication involves analyzing captured packets for specific encryption-related HCI commands and events. The script utilizes `pyshark` to process the captured packets and identify encryption indicators.

Code Overview for Encryption Verification

Imports and Function Definitions

python
Copy code

```
import pyshark
```

1.

Checking for Encryption The `check_encryption` function analyzes the captured packets to determine if encryption is being used in the communication.

python
Copy code

```
def check_encryption(pcapng_file):
    encrypted_communication = False
    protocols_involved = set()

    try:
        cap = pyshark.FileCapture(pcapng_file,
display_filter='bthci_evt || bthci_cmd || btl2cap')

        for packet in cap:
            if hasattr(packet, 'bthci_cmd') and
packet.bthci_cmd.opcode == '0x01':
                encrypted_communication = True
                protocols_involved.add('bthci_cmd')
                print(f"Encryption-related HCI command (0x01) found in
packet number: {packet.number}")
```

```
            elif hasattr(packet, 'bthci_evt') and
packet.bthci_evt.code == '0x08':
                encrypted_communication = True
                protocols_involved.add('bthci_evt')
                print(f"Encryption Change Event (0x08) found in packet
number: {packet.number}")
            elif hasattr(packet, 'btl2cap') and
packet.btl2cap.channel_id == '0x0001':
                encrypted_communication = True
                protocols_involved.add('btl2cap')
                print(f"L2CAP Signaling Channel found in packet
number: {packet.number}")
            elif hasattr(packet, 'btl2cap') and packet.btl2cap.control
!= '0x00':
                encrypted_communication = True
                protocols_involved.add('btl2cap')
                print(f"L2CAP encrypted data found in packet number:
{packet.number}")

    except Exception as e:
        print(f"An error occurred: {e}")

    cap.close()
    return encrypted_communication, protocols_involved
```

2.

Explanation

- **HCI Commands and Events**:
  - `bthci_cmd.opcode == '0x01'`: Indicates an encryption-related command was sent.
  - `bthci_evt.code == '0x08'`: Indicates an Encryption Change event, confirming encryption status.
- **L2CAP Analysis**:
  - `btl2cap.channel_id == '0x0001'`: Identifies the L2CAP signaling channel.
  - `btl2cap.control != '0x00'`: Detects encrypted data on the L2CAP channel.

The `check_encryption` function processes each packet to identify these specific indicators, thus confirming the presence of encryption in the Bluetooth communication.

Conclusion

By analyzing the captured packets for specific HCI commands and L2CAP indicators, the script effectively verifies if the Bluetooth communication is encrypted. This method ensures secure communication between Bluetooth devices by confirming encryption status through packet analysis.

References

- Bluetooth Core Specification: HCI Commands and Events
- `pyshark` Documentation

This approach can be expanded to include additional security checks and enhancements to further ensure the robustness and security of Bluetooth communications