Ghost Tunnel
Covert Data Exfiltration Channel to Circumvent Air Gapping

Hongjian CAO, Kunzhe CHAI, Jun LI
PegasusTeam, 360 Security Technology
April 12, 2018
Who We Are

360 Security Technology is a leading Internet security company in Asia. Our core products are anti-virus security software for PC and cellphones.

PegasusTeam was founded in 2015. we focus on the wireless security and wireless penetration testing.
Agenda

• Introduction
• Previous research on Air-Gapped attack
• Ghost Tunnel Introduction
• Ghost Tunnel implementation
• demo
Introduction

- Air-Gapping
- Attack events
Air Gapping

- Air gapping
  - Wikipedia: “air gapping[1] is a network security measure employed on one or more computers to ensure that a secure computer network is physically isolated from unsecured networks, such as the public Internet or an unsecured local area network.[2] The name arises from the technique of creating a network that is physically separated (with a conceptual air gap) from all other networks.”

- Air gapping aims to avoid the intrusion and data leakage through network connections
Air-Gapped Network

- Considered to be the most secure
Nothing Is Impossible

- Attack Vectors
  - Malicious USB
  - Employee's laptop
Stuxnet Worm (2010)

- Attacking initiated via an infected USB drive
- Designed to sabotage centrifuges used at a uranium enrichment plant in Iran
NSA Leaks (2013)

- COTTONMOUTH-I
  - A USB hardware implant
  - Air-Gap bridging
  - Extracting data from targeted systems via RF signals
Previous research on Air-Gapped attacks
Previous research - 1

• Using radio frequencies to transmit data from a computer
  - Computer monitor
  - Mobile phone FM radio receiver

url: https://thehackernews.com/2014/10/airhopper-hacking-into-isolated.html
Previous research - 2

• A covert bi-directional communication channel between two close by air-gapped computers communicating via heat

url: https://thehackernews.com/2015/03/hacking-air-gapped-computer.html
Previous research - 3

- Data exfiltration via RF signal by attacking Siemens PLCs

Ghost Tunnel

A Covert Data Exfiltration Channel Using WiFi
Air-gapped Attack

• Implant
  - Malicious software/hardware

• A covert communication channel
  - Any medium that can carry data is possible
Ghost Tunnel

Implant malware
• USB HID attack
• BashBunny

Setup C&C tunnel
• Via 802.11 beacon and probe request & response

Exfiltrate data
• Execute Command
Ghost Tunnel

• Can bypass firewalls
• Cross-Platform support
• Allow up to 256 clients
• Effective range up to 50 meters
The Usual Wifi Connection Process

1. Beacon (Broadcast SSID)
2. Probe Request
3. Probe Response
4. Authentication Request
5. Authentication Response
6. Association Request
7. Association Response

WiFi Connection Established
Ghost Tunnel – No WiFi Connection

1. Beacon (Broadcast SSID)
2. Probe Request
3. Probe Response
4. Authentication Request
5. Authentication Response
6. Association Request
7. Association Response

Ghost Tunnel Connection

WiFi Connection Established
802.11 State

State 3
Authenticated and associated

State 2
Authenticated and unassociated

State 1
Unauthenticated and unassociated

Class 1, 2, and 3 frames

Disassociation
Association

Class 1 and 2 frames

Deauthentication
Authentication

Class 1 frames

802.11 State Diagram
## Class 1 Frames

<table>
<thead>
<tr>
<th>Control</th>
<th>Management</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>Probe Request</td>
<td>Frame w/DS bits false</td>
</tr>
<tr>
<td>CTS</td>
<td>Probe Response</td>
<td></td>
</tr>
<tr>
<td>Ack</td>
<td>Beacon</td>
<td></td>
</tr>
<tr>
<td>CF-End</td>
<td>Authentication</td>
<td></td>
</tr>
<tr>
<td>CF-End + CF-Ack</td>
<td>Deauthentication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATIM</td>
<td></td>
</tr>
</tbody>
</table>
Scanning for Wifi Networks
Ghost Tunnel – No WiFi Connection

• A covert WiFi channel using Beacon, Probe Request, Probe Response

• A special SSID as the identifier
Ghost Tunnel Implementation
### 802.11 Frame

- **Control frame**
- **Management frame**
- **Data frame**

<table>
<thead>
<tr>
<th>Octets: 2</th>
<th>2</th>
<th>6</th>
<th>0 or 6</th>
<th>0 or 6</th>
<th>0 or 2</th>
<th>0 or 6</th>
<th>0 or 2</th>
<th>0 or 4</th>
<th>variable</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Control</td>
<td>Duration /ID</td>
<td>Address 1</td>
<td>Address 2</td>
<td>Address 3</td>
<td>Sequence Control</td>
<td>Address 4</td>
<td>QoS Control</td>
<td>HT Control</td>
<td>Frame Body</td>
<td>FCS</td>
</tr>
</tbody>
</table>

**Frame header**
802.11 Management Frame Body

- **Management Frame Body**
  - Fields
  - Information Elements
The components of Information Element

- Element ID: 1 Byte
- Length: 1 Byte
- Information: 0-255 Bytes
  - SSID
  - Vendor Specific

<table>
<thead>
<tr>
<th>Octets:</th>
<th>1</th>
<th>1</th>
<th>variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element ID</td>
<td>Length</td>
<td>Information(payload)</td>
<td></td>
</tr>
</tbody>
</table>

Element Format
SSID Element

- Identity of an ESS or IBSS
- SSID length 0-32 Bytes

<table>
<thead>
<tr>
<th>Octets:</th>
<th>1</th>
<th>1</th>
<th>0-32</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Element ID</td>
<td>Length</td>
<td>SSID(Payload)</td>
</tr>
</tbody>
</table>
Vendor Specific Element

- ID = 221
- Organization Identifier
- Vendor-Specific content

<table>
<thead>
<tr>
<th>Octets:</th>
<th>Element ID</th>
<th>Length</th>
<th>Organization Identifier</th>
<th>Vendor-specific content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3 or 5</td>
<td>variable</td>
<td>variable</td>
</tr>
</tbody>
</table>

Payload
Key Problem

- How to send and receive 802.11 data frames through local wireless network interface in user space?

- Wireless network interface mode
  - Master (Acting as an AP)
  - Managed (Station)
  - Monitor (Monitor all traffic)
  - ...

Through Operating System WiFi API

- **Windows**
  - Native Wifi API
- **Mac OS X**
  - coreWLAN
- **Linux**
  - nl80211 & libnl
Windows Client: Send And Receive

DWORD WINAPI WlanScan(
    _In_ HANDLE hClientHandle,
    _In_ const GUID *pInterfaceGuid,
    _In_opt_ const PDOT11_SSID *pDot11Ssid,
    _In_opt_ const PWLAN_RAW_DATA *pIeData,
    _Reserved_ PVOID pReserved);

• scan for available wireless networks
  - pDot11Ssid, specifies the SSID of the network to be scanned
  - pIeData != NULL, send probe request
  - pIeData == NULL, not send probe request
Packet payload Format

- **DOT11SSID**
  - Contains the SSID
  - The maximum length is 32

```c
typedef struct _DOT11SSID {
    ULONG uSSIDLength;
    UCHAR ucSSID[DOT11SSID_MAX_LENGTH];
} DOT11SSID, *PDOT11SSID;
```

- **WLAN_RAW_DATA**
  - Contains the elements data
  - Not exceed 240 bytes

```c
typedef struct _WLAN_RAW_DATA {
    DWORD dwDataSize;
    BYTE DataBlob[1];
} WLAN_RAW_DATA, *PWLAN_RAW_DATA;
```
Windows Client : Receive

DWORD WINAPI WlanGetNetworkBssList(
  _In_ HANDLE hClientHandle,
  _In_ const GUID *pInterfaceGuid,
  const PDOT11_SSID pDot11Ssid,
  _In_ DOT11_BSS_TYPE dot11BssType,
  _In_ BOOL bSecurityEnabled,
  _Reserved_ PVOID pReserved,
  _Out_ PWLAN_BSS_LIST *ppWlanBssList);

• Retrieve available wireless networks list
• ppWlanBssList
  - Receive the returned list of BSS entries
Windows Client: Receive

- **WLAN_BSS_LIST**
  - An array of WLAN_BSS_ENTRY structures that contains information about a network

```
WLAN_BSS_LIST

<table>
<thead>
<tr>
<th>dwTotalSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwNumberOfItems</td>
</tr>
<tr>
<td>wlanBssEntries</td>
</tr>
</tbody>
</table>

WLAN_BSS_ENTRY

| dot11Ssid |
| uPhyId |
| dot11Bssid |
| ... |
| ulleOffset |
| ulleSize |

Payloads

| IE [0] |
| IE [1] |
| ... |
| IE [221] |
```
Mac Client : Send

- CWInterface
  - func scanForNetworks(withSSID: Data?)
Mac Client : Receive

- CWInterface
  - func scanForNetworks(withSSID: Data?)
  - func cachedScanResults() -> Set<CWNetwork>?

- CWNetwork
  - informationElementData: Data?
C&C Server: Send And Receive

- Modified hostapd and hostapd_cli
- USB WiFi card
Ghost Tunnel

360PegasusTeam
Thanks!

Any questions?