Network Protocol Reversing (Intro)

Offensive Network Security
Florida State University
Spring 2014

Outline

- Why?
- Where?
- What?
- Tools
- Definitions
- Techniques

Why?

- Data hidden in known protocols
- Poorly documented network protocol
- Undocumented/Unknown protocol
- Think: botnet C&C
- Incident response with captured network data and no binaries
  - What if we have the client?
  - What if we have the server?

What is a Protocol?

- Protocol -- A system of rules that explain correct conduct and procedures to be followed in formal situations (Merriam-Webster)
- Communications Protocol -- Digital rules for exchange of data between computers (Wikipedia)
- Network Protocol -- A system of rules designated when data is exchange via a computer network (Wikipedia)

Where are Protocols?

- Conventional known Protocols
  - IP
  - TCP / UDP
  - HTTP / DNS / DHCP / IRC
- How can an unknown protocol exist in the above known protocols?
- Covert channels
- Steganographic Channels
- Should a responder trust any protocols when called to a possible data theft when searching through captured network traffic?

What are Protocols?

- A new protocol can be implemented in Application Layer
- What are the fields?
- Is it encrypted?
- Are there any defined patterns?

What is Protocol Reversing?

- Determine the fields, structure, and communication of a network protocol without any knowledge of the protocol under investigation
- Combine knowledge from captured data, traces from binaries to determine network protocol

Tools

- ltrace, strace, dtrace -- determine function and system calls
- hexdump, hexedit -- look at the network data bytes
- ncat, socat -- communicate with clients/server without coding
- Python (or another language) -- code the network communication
- Proxy servers -- buffer/redirect network communication
- Canape -- Network testing tool for arbitrary network protocols
- Protocol Debugger -- Test and manipulate stateful protocols
- Discoverer -- Cluster messages together to determine protocol
- NetZob -- Use Bioinformatics to determine protocol
- Not an exhaustive list

Wireshark or tcpdump?

- How are the following generated packets dissected in Wireshark/tcpdump?

pkt = Ether(type=0x0800)/ARP()

pkt = Ether(type=0x0806)/IP()/IP()

pkt = Ether()/IP(proto='udp')/TCP()

pkt = Ether()/IP(proto='tcp')/UDP()

pkt = Ether()/IP()/TCP()/"GET / HTTP/1.1\r\nhost:remote.edu\r\n\r\n"

Bytes on a Wire

- Network communication “convention” transfers bytes in big endian format
  - int
  - short
  - double
- Does an attacker need to obey this “convention”?
- ‘data’ = ‘\x64\x61\x74\x61’
- int(0x64617461) = 1684108385
- short(0x6461) = 25697
- short(0x7461) = 29793
- bin(0x64617461) = 0b11001000110000101110100011000 01
  - Could these bytes represent flags for a protocol?
- What other data transformations could exist?

Tracing Function Calls

- If client is available
- If server is available
- Maybe malware server is dead, need to see whom it is trying to connect
- Use ltrace, strace, dtrace for capturing system or library calls
- Can determine if DNS call is made, socket connections, etc.

Tracing Function Calls

```
> ltrace -S ssh 192.168.2.1
.. .. ..
SYS_socket(16, 3, 0, 0x7f00d245e100)                                    = 3
SYS_bind(3, 0x7ffffe524200, 12, 0x7f00d245e100)                         = 0
SYS_getsockname(3, 0x7ffffe524200, 0x7ffffe5241fc, 0x7f00d245e100)                  = 0
SYS_sendto(3, 0x7ffffe524150, 20, 0)                                    = 20
SYS_recvmsg(3, 0x7ffffe524170, 0, 0)                                    = 168
SYS_recvmsg(3, 0x7ffffe524170, 0, 0)                                    = 192
SYS_recvmsg(3, 0x7ffffe524170, 0, 0x8000)                               = 20
SYS_socket(1, 0x80801, 0, 168)                                          = 4
SYS_connect(4, 0x7ffffe522d40, 110, 168)                                = -2
SYS_close(4)                                                            = 0
SYS_close(3)                                                            = 0
<... getaddrinfo resumed> )                                             = 0
getnameinfo(0x7f00d6347390, 16, "", 1025, "22", 32, 3)                  = 0
__errno_location()                                                      = 0x7f00d4af96b0
socket(2, 1, 6 <unfinished ...>
SYS_socket(2, 1, 6, 64)                                                 = 3
<... socket resumed> )                                                  = 3
fcntl(3, 2, 1, -1 <unfinished ...>
SYS_fcntl(3, 2, 1, 64)                                                  = 0
<... fcntl resumed> )                                                   = 0
gettimeofday(0x7ffffe524520, 0)                                         = 0
connect(3, 0x7f00d6347390, 16, 1 <unfinished ...>
SYS_connect(3, 0x7f00d6347390, 16, 64
```

Network Function Calls

- socket -- create an endpoint for network communication
- bind -- Assign an address (name) to a socket
- listen -- Wait for connections on a socket
- accept -- Accept for incoming connection in queue from a socket
- connect -- Start a connection on a socket
- send --Transmit data to receiver in a **connected** state
  - sendmsg, sendto -- Transmit data for any connection
- recv -- Receive data from a socket in any connection state
  - recvmsg, recvfrom -- Provides same functionality
- shutdown -- Stop read, write, or both of full-duplex connection
- close -- close file descriptor (socket identifier)

Other Function Calls

- gethostbyname -- Translate domain name to host information
- gethostbyaddr -- Translate binary IP to host information
- getaddrinfo/freeaddrinfo -- Obtain IP/port numbers for hostname or service
- memcmp, strcmp, strncmp, etc. -- Compare bytes / strings
- memset, memcpy, strcpy, strncpy, etc -- Copy bytes / strings
- strlen -- length of a string
- htons, ntohs, htonl, ntohl -- Swap host/network byte order (16bits/32bits)
- inet_ntoa, inet_aton, inet_pton, etc. -- IP address manipulation functions

Resources

- http://capec.mitre.org/data/definitions/192.html
- http://www.cs.berkeley.edu/~dawnsong/papers/2012%20Automatic%20Protocol%20Reverse%20Engineering.pdf
- http://digital-forensics.sans.org/blog/2012/07/03/an-overview-of-protocol-reverse-engineering
- http://www.matasano.com/research/BH-US-06-Rauch.pdf
- http://research.microsoft.com/en-us/um/people/helenw/papers/discoverer.pdf