STEFFEN WENDZEL

Protocol Channels

Difficulty

Covert channel techniques are used by attackers to transfer hidden data. There are two main categories of covert channels: timing channels and storage channels. This text introduces a new storage channel technique called protocol channels.

WHAT YOU WILL LEARN...

The basics of network covert channels

How protocol channels work and how one can use them

WHAT YOU SHOULD KNOW...

Basics of Covert Channels (optional)

Basics of TCP/IP.

A protocol channel switches one of at least two protocols to send a bit combination to a destination. The main goal of a protocol channel is that the packets sent look equal to all other usual packets of the system what makes a protocol channel hard to detect.

Introduction

For attackers it is usual to transfer different kinds of hidden information trough hacked or public networks. The solution for this task can be to use a so called covert channel technique like they are known since many years.

A new storage channel technique I call *protocol channel* includes hidden information only in the header part of protocols that specify an encapsulated protocol (e.g. the field *Ether Type* in Ethernet – Table 1 lists more of such protocol header parts). For example: If a protocol channel would use ICMP and ARP, while ICMP means that a *0* bit was transfered and ARP means that a *1* bit was transfered, then the packet combination sent to transfer the bit combination *0011* would be ICMP, ICMP, ARP, ARP. This sounds easy but there are two important things to mention:

- A protocol channel may not contain any other information that identifies the channel nor other hidden information because this would make the protocol channel much easier to detect. A typical packet would be a HTTP-Request seen hundreds of times each day in a typical network. The HTTP-Header would not include any kind of hidden information itself. The payload may also be free of any hidden information.
- It is also important that a protocol channel only uses *usual* protocols of the given network since protocols *unusual* for the network would be easy to detect. An interesting algorithm to identify such protocols for *adaptive covert channels* (I call them *protocol hopping covert channels* and invented them earlier) was introduced by [YADALI08].

The higher the number of available protocols for a protocol channel is, the higher amount of information can be transferred within one packet since more states are available. Given the above example, 2 different states are available, which represents 1 bit. If the attacker could use 4 different protocols, a packet would represent 2 bits. Figure 1 shows a sample protocol channel using 4 different protocols where each packet represents 2 bits of covert information.

This does not allow high covert channel bandwidths but is more than enough to transfer sniffed passwords or other tiny information. The need for a high bandwidth decreases dramatically if the attacker uses some

compressing algorithm (like modify an ASCII text by converting it to a 6 bit representation of the most printable characters). The proof of concept code *pct* uses a minimalized 5 bit ASCII encoding and a 6th bit as a parity bit. You can find pct on the Hakin9 website.

Proof of Concept Implementation

There is a tiny proof of concept implementation for Linux 2.6 called *pct* (protocol channel tool) available. As already mentioned, pct uses a 5 bit ASCII encoding and adds a 6th parity bit.

This is possible because most unprintable characters are not included here. All lower case characters are made upper case. Digits are also not available since it is possible to write them as text (*ONE* instead of *1*).

pct uses the Perl modules CPAN/Net::RawIP and CPAN/Net::ARP and is based on ARP and ICMP packets while ARP has the meaning of a zero bit and ICMP packets have the meaning of a *1* bit. Due to the fact that ARP is used, the proof of concept code can only transfer hidden information within a subnet.

After a typical break in the attacker could use pct to stay hidden while transferring secret (stolen) data using the pct protocol channel. It is also possible that an attacker could use pct to send hidden information into a hacked network to control bots which are part of a botnet.

To use pct, one first has to start the receiving component called *pct_receiver*. It takes the network device to listen on as a parameter (e.g. `eth0` or *lo*). Listing 1 shows how to do that.

The sending component `pct_sender` takes more parameters: (1) The network interface to send from, (2) the source IP address, (3) the destination IP address, (4) the source MAC address, (5) the destination MAC address, (6) the initial value for the ICMP sequence number (e.g. 0x053c) of the ICMP echo packets and (7) the secret message to send trough the protocol channel. Listing 2 shows an example call of the program.

Protocol Hopping Covert Channels

If you already know *Protocol Hopping Covert Channels* then you may ask what the difference is between */these/* channels and protocol channels. The main difference is that protocol channels modify no information of a network packet excluding the protocol identifier of the encapsulated protocol.

Table 1. *Parts of Headers used to include Protocol Channel information*

Layer / Protocol	Used Part of the Header
Network Access Layer / Ethernet	Ether Type
Network Access Layer / PPP	Protocol
Internet Layer / Ipv4	Protocol
Internet Layer / Ipv6	Next Header
Transport Layer / TCP and UDP	(Source and) Destination Port

What are Covert Channels?

A definition of *covert channels* can be found in [BISHOP06]: *A covert channel is a path of communication that was not designed to be used for communication*. He also defines the difference between the two main categories of covert channels: *A covert storage channel uses an attribute of the shared resource. A covert timing channel uses a temporal or ordering relationship among accesses to a shared resource.* The TCSEC standard includes a similar definition: *Covert storage channels include all vehicles that would allow the direct or indirect writing of a storage location by one process and the direct or indirect reading of it by another. Covert timing channels include all vehicles that would allow one process to signal information to another process by modulating its own use of system resources in such a way that the change in response time observed by the second process would provide information.* [DOD85].

A covert channel that changes an attribute of a HTTP header (e.g. the cookie field to include hidden information in the cookie value) would be a storage channel. Instead the covert channel could measure the manipulated response time for HTTP requests what whould be a typical timing channel.

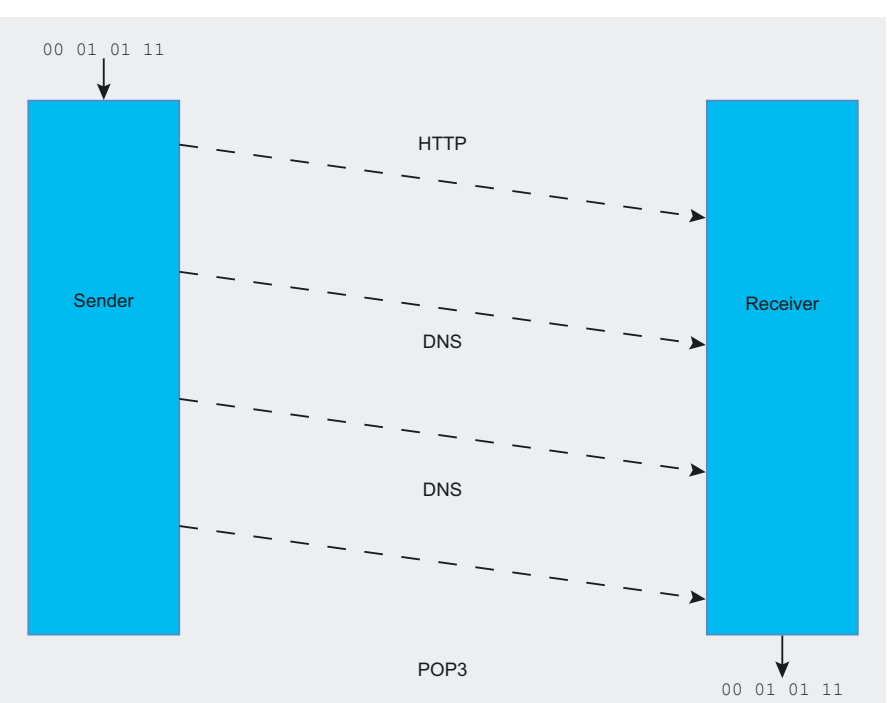

Figure 1. *A protocol channel transfering bits using a set of 4 different protocols*

This makes protocol channels much harder to detect than protocol hopping covert channels. Following the definition of a protocol hopping covert channel, a protocol channel could be defined as a special kind of a protocol hopping covert channel. The next subsection shows some problems related to this difference.

Problems

Since a protocol channel only contains one or two (usually not more) bits of hidden information per packet, it is not possible to include reliability information (like an ACK flag or a sequence number) in such a packet. If a normal packet that doesn't belong to the protocol channel would be accepted by the receiver of a protocol channel, the whole channel would be de-synchronized. It is not possibly to identify packets which (not) belong to the protocol channel if they use one of the protocols the protocol channel uses.

This lack of a *micro protocol* (a covert protocol that includes meta information for the transferred covert data) that implements reliability and a identification information is also one of the mayor differences between protocol channels and protocol hopping covert channels.

Another problem is the fragmentation as well as the loss of packets. If a packet was de-fragmented, the receiver would receive it two times what means that the bit combination of the received packet would be used twice what would result in a destroyed bit sequence on the receiving system.

The channel would end up de-synchronized in this case too. A receiver could check for packets that include the *More Fragments* flag of IPv4 as a solution for this problem. Lost packets create a hole in the bit combination what results in the same de-synchronization problem.

Listing 1. *Start of pct_reciever*

```
$ sudo ./pct_receiver eth0
RECEIVING MESSAGES -
PRESS CTRL-C TO FINISH
```

Listing 2. *Using pct_sender.pl to send the String "HELLO"*

```
$ sudo perl ./pct_sender.pl eth0 \
192.168.2.22 192.168.2.21 \
 00:1d:09:35:87:c4 \
 00:17:31:23:9c:43 0x053c \

 "HELLO''

sending payload[0]=H
sending=00111
sending bit 0=0 ARP
sending bit 1=0 ARP
sending bit 2=1 ICMP
sending bit 3=1 ICMP
sending bit 4=1 ICMP
Seqnr now=1343
...
```

What are Protocol Hopping Covert Channels

Protocol Hopping Covert Channels are storage channels that have a set of at least two different network protocols to use. They switch their used protocol while transferring secret information. For example: They use the HTTP cookie value as well as a POP3 message number to hide data in. Then a first packet could be an HTTP request and a second packet could be a POP3 RETR command to send such information. The next packet could be HTTP (or POP3) again. If one of the protocols used is detected, the other protocol is still undetected. This makes a forensic analysis much harder. I described Protocol Hopping Covert Channels in more detail in [WEND07].

Conclusion

Protocol channels provide attackers a new way to stay hidden in networks. Even if a detection by network security monitoring systems is possible – e.g. because of unusual protocols used by the attacker – a regeneration of the hidden data is as good as impossible since it would need information about the transferred data type, the way the sent protocol combinations are interpreted (big-endian or little-endian) and a recording of all sent packets to make a regeneration possible.

Due to this fact, protocol channels are much harder to detect than protocol hopping covert channels but also are less stable and provide a lower bandwidth.

References

- [BISHOP06] Bishop, M.: Computer Security: Art and Science, Addison-Wesley, 9th Printing, October 2006.
- [DOD85] Department of Defence: Trusted Computer System Evaluation Criteria (TCSEC, DoD 5200.28-STD), 1985. URL: *http://csrc.nist.gov/publications/history/dod85.pdf*
- [WEND07] Wendzel, S.: Protocol Hopping Covert Channels. An Idea and the Implementation of a Protocol switching covert channel. URL: *http://www.wendzel.de/?sub=paper_phcc. A german text about this topic can be found in Hakin9 03/08.*
- [YADALI08] F. Yarochkin, S.-Y. Dai, C.-H. Lin, Y. Huang, S.-Y. Kuo: Towards Adaptive Covert Communication System, Dep. of Electrical Engineering, National Taiwan University, 2008.

Steffen Wendzel

Steffen Wendzel is author of two German books about Linux and wrote also one about Network Security. He is about to finish his diploma degree in computer science at the Kempten University of Applied Sciences, Germany.