Global Information Assurance Certification Paper

GCIA Practical

Intrusion Detection In Depth

Joseph R. Taylor
Version 2.9x
October 4, 2001

Assignment 1 – "Describe the State of Intrusion Detection"

Intrusion Detection In Depth
GCIA Practical Assignment
Joseph Taylor
Version 2.9x

The RPC Protocol – IDS Evasion and Denial of Service Using RPC Design Flaws

Joseph R. Taylor

IDS Research Engineer

Enterasys Networks

[First Edition: February 21, 2001]

[Second Edition: October 4, 2001]

Overview

Robert Graham's release of Sidestep in early February 2001 identified basic Intrusion Detection System (IDS) evasion techniques in the BackOrifice application, DNS (Domain Name Service), FTP (File Transfer Protocol), HTTP (Hyper-Text Transfer Protocol), RPC (Remote Procedure Call), and SNMP (Simple Network Management Protocol). Sidestep demonstrates only one method of evading signature-based IDSes using the RPC protocol. This paper goes deeper into RPC to show how it operates, its inherent weaknesses in terms of network security, a full disclosure of all known evasion techniques, and solution sets to counter these weaknesses for both network-level security and IDS technology.

We will present this information "in plain English" first – the technical details will be included in the Appendices.

The key points of this discussion are:

- There are an infinite number of possible ways to evade IDSes using the weaknesses inherent in the RPC protocol.
- One evasion method constitutes a Denial-of-Service (DoS) attack against RPC servers and scales easily and rapidly into a full Distributed Denial-of-Service (DDoS) attack. This evasion method also contains DoS and DDoS implications for all forms of IDSes.
- Solution sets from both network-security and IDS technology perspectives are, fortunately, relatively simple to implement and effective.
- The relative threat severity of these attacks is, at most, medium.

The RPC Protocol "In Plain English"

What the RPC Protocol Does

According to the Merriam-Webster Dictionary, a protocol is "a set of conventions governing the treatment and especially the formatting of data in an electronic communications system". Put simply, a protocol defines the rules a client must follow to communicate with a server (and vice-versa) for a particular type of network service offering.

The chain of events for the RPC protocol begins in OSI Layer 5 (Session Layer) with the core code of the RPC communication conventions. At OSI Layer 6 (Presentation), XDR (eXternal Data Representation) encodes the input to and decodes output from the portmap/rpcbind server at OSI Layer 7 (Application). For the purposes of this paper, portmap and rpcbind are equivalent services with different names because they run on different versions of UNIX. They both run on TCP and UDP port 111 and/or 32771 through 32779. You can think of portmap/rpcbind as a phone book that RPC-based programs must use to talk with each other. The portmap (Linux, xBSD) / rpcbind (Solaris) server's purpose is to act as a central registration facility for all other RPC-based services, such as NFS (Network File System) and NIS (Network Information System). When these services start up, they register the ports they will be listening on with portmap/rpcbind. When an RPC call comes in from another application or service, it must ask portmap/rpcbind for the port of the service it wants to talk with. When portmap/rpcbind returns this information to the caller, the caller will use it to begin direct communication with the service it was seeking. The evasion and attack techniques we will discuss target portmap/rpcbind.

A Normal RPC Request-for-Information "Conversation"

The *rpcinfo* utility is provided in UNIX for querying portmap/rpcbind for information it has about the RPC-based services registered with it. The *–p* option to rpcinfo asks portmap/rpcbind to return all of the information available about every service registered with it. *rpcinfo –p* is often used by hackers or system crackers to gather information about what RPC services are offered by a given host so that they can decide which, if any, RPC-based attack methods they have at the ready will apply to the targeted host.

A normal *rpcinfo –p* command equates to the following English conversation:

rpcinfo: "This is the last fragment of data I have to send you. The data fragment is 40 bytes long. I want to use this tracking number to keep our conversation synchronized. I am placing a call message. I am using RPC Version number 2. I need to speak to portmap/rpcbind and I expect its program version number to also be 2. I am asking portmap/rpcbind to give all the information it has about every registered service on the local host. I am not using any authentication or verification mechanism."

portmap/rpcbind: "Because the data you are sending me represents the last fragment of data you have to send I can begin processing it. This is the last fragment of data I will be sending to you and it is 988 bytes in length. I acknowledge your conversation tracking number by sending it back to you. I am sending a reply message to your request. Because your input was formatted properly I am accepting your call message. Because you presented me no authentication or verification data, I won't use any, either. Your call message executed successfully. I have the information you requested. Each piece of information will be of different lengths. When I have completed sending you all of the information I have, I will tell you that there is no information left to send and our conversation will be concluded."

A *Sidestep* RPC Request-for-Information "Conversation"

The *Sidestep* program also performs an rpcinfo –p request, but by a different method that attempts to evade IDSes by changing the pattern of the conversation.

Sidestep: "This is not the last fragment of data I have to send you. This data fragment is one byte long. Here is that data byte. I will repeat this message 39 more times…

portmap/rpcbind: "Because the data you are sending me does not represent the last fragment of data you have to send, I will not be able to begin processing your request until you tell me that you have sent the last data fragment…

Sidestep: ..this is the last data fragment. It is one byte long. Here is that byte. My message is complete."

portmap/rpcbind: ..it took a little longer than usual, but I now have all of the data you sent me, so I can begin processing it. This is the last fragment of data I will be sending to you and it is 988 bytes in length. I acknowledge your conversation tracking number by sending it back to you. I am sending a reply message to your request. Because your input was formatted properly I am accepting your call message. Because you presented me no authentication or verification data, I won't use any, either. Your call message executed successfully. I have the information you requested. Each piece of information will be of different lengths. When I have completed sending you all of the information I have, I will tell you that there is no information left to send and our conversation will be concluded."

Beyond *Sidestep*

The conversation pattern used by *Sidestep* is static from one invocation to the next. That fact allows a signature-based IDS to identify its technique and generate an alert. Unfortunately, the static technique used by *Sidestep* is not the only one possible. Here is a variation on the evasion theme.

Hacked-up RPC Client: "This is not the last fragment of data I have to send you. This data fragment is seven bytes long. Here are those bytes. This is not the last fragment of data I have to send you. This data fragment is three bytes long. Here are those bytes. This is not the last fragment of data I have to send you. This data fragment is five bytes long. Here are those bytes. This is not the last fragment of data I have to send you. This data fragment is nine bytes long. Here are those bytes. This is not the last fragment of data I have to send you. This data fragment is two bytes long.

Here are those bytes. This is not the last fragment of data I have to send you. This data fragment is six bytes long. Here are those bytes. This is not the last fragment of data I have to send you. This data fragment is seven bytes long. Here are those bytes. This is the last data fragment. It is one byte long. Here is that byte. My message is complete."

portmap/rpcbind: "Because the data you are sending me does not represent the last fragment of data you have to send, I will not be able to begin processing your request until you tell me that you have sent the last data fragment…

portmap/rpcbind: ..it took a little longer than usual, but I now have all of the data you sent me, so I can begin processing it. This is the last fragment of data I will be sending to you and it is 988 bytes in length. I acknowledge your conversation tracking number by sending it back to you. I am sending a reply message to your request. Because your input was formatted properly I am accepting your call message. Because you presented me no authentication or verification data, I won't use any, either. Your call message executed successfully. I have the information you requested. Each piece of information will be of different lengths. When I have completed sending you all of the information I have, I will tell you that there is no information left to send and our conversation will be concluded."

From the *Sidestep* conversation and a variant, it becomes clear that any conversation (evasion) pattern can be created, provided the following rules are followed:

1. All fragment message byte values used must add up to 40.

2. All data bytes must be encoded properly in the message.

3. All data fragment messages except the last must indicate that more data fragments follow and must properly indicate the number of data bytes that follow.

4. A single last fragment message must be sent with valid length and associated data bytes.

Item (1) above implies that the total numbers of possible conversation variations are 40! or 8.159×10^{47}. A signature-based IDS cannot possibly cope with such a large number of signatures even if one could write them all. That

forces signature-based IDSes to incorporate a RPC protocol decoding algorithm to extract the "normal conversation" and run that result through

signature-matching analysis. We have observed that the minimum number of bytes for a request-for-information conversation (a normal rpcinfo -p command) is 44 bytes of data. *Sidestep* uses the maximum number of bytes for its static variant, which comes in at 200 bytes. All conversation variants will use no less than 44 bytes and no more than 200 bytes. From a decoding standpoint, this is not a lot of data to manipulate, so the IDS RPC protocol decoder should not incur a significant additional workload. Unfortunately, there is one conversation technique that is toxic to both portmap/rpcbind and IDSes that employ RPC protocol decoding.

The Toxic RPC Request-for-Information "Conversation" a.k.a. "Null-Byte Encoding"

In the preceding discussions, we have always assumed a set of logical conditions for RPC fragment messages – that we were or were not sending the last fragment of data and that the data bytes had some length greater than zero. What happens when you set data length to zero? Here's what that conversation looks like – think Jack Nicholson in *The Shining* ;-)

Hacked-up RPC Client: "This is not the last fragment of data I have to send you. This data fragment is zero bytes long. This is not the last fragment of data I have to send you. This data fragment is zero bytes long. This is not the last fragment of data I have to send you. This data fragment is zero bytes long…

(Millions of such messages later)

This is not the last fragment of data I have to send you. This data fragment is zero bytes long. This is not the last fragment of data I have to send you. This data fragment is zero bytes long. This is not the last fragment of data I have to send you. This data fragment is zero bytes long. This is not the last fragment of data I have to send you. This data fragment is zero bytes long….

(While this is happening, portmap/rpcbind is thinking…)

portmap/rpcbind: "I cannot do anything but wait until you notify me that the last data fragment has been sent. I cannot talk to anyone else until your conversation has been completed."

The conversation constitutes a denial-of-service attack against both the remote portmap/rpcbind server and the IDS that is attempting to decode the conversation. In our experiments with "null-byte" conversations, we have found that portmap/rpcbind will deny connections to any other RPC-based application that needs its services while the toxic conversation is taking place. We have not found that portmap/rpcbind will crash, but we have not pushed the limits very hard. The largest single null-byte encoded stream we have tried came in at 400 million bytes. This attack denied service to or from portmap/rpcbind for about 10 minutes. If you think of this conversation as a module that can plug in to DDoS programs like TFN, Trin00, or Stacheldraht, you'll see that the null-byte technique quickly and easily scales out to a network-spanning DoS attack that could consume significant bandwidth. In single-host DoS and network DDoS cases, the IDSes attempting protocol decode on this technique are likely to be overwhelmed.

Further, we have found that legitimate queries and evasions can be embedded in null-byte message streams. If the query or evasion is formed correctly, portmap/rpcbind will respond eventually. The embedding can be inserted as either a single "chunk", or it can be interspersed throughout the null encoded stream. That capability increases the number of possible evasions to infinity, for all practical purposes.

Observations, Solutions, etc.

- **Observation:** The core problem with RPC hinges on its message fragmentation feature. This is called the "Last Fragment/Fragment Length" message (or LF/FL for short). portmap/rpcbind will not begin processing a request until it know it has its last data fragment. Null-Byte RPC encoding (Last Fragment set to No and Fragment Length of Zero) appears to be an oversight on the part of the developers of the RPC protocol. This oversight allows denial-of-service attacks against individual hosts offering RPC services and scales out to network DDoS bandwidth-consumption attacks.

  Solution(s): There are, fortunately, a lot of good ways for networks to mitigate against null-byte RPC flooding:

  - Block TCP port 111 and 32771 through 32779 (portmap/rpcbind) at the network head-end. We would also suggest a careful analysis of internal-to-internal network requirements – where TCP ports 111 and 32771 through 32779 are not required for intranetwork communications, block them on internal routers as well.

  - Deploy Wietse Venema's replacement portmap/rpcbind on all UNIX hosts offering RPC services. When Wietse's replacement portmapper is used, inbound null-byte streams are killed almost immediately if the calling host is not authorized to make the connection.

  - The RPCSEC_GSS (RFC 2203) credentialing and authentication mechanism appears to hold some promise, but we have not seen an implementation of its features in an active network. It is not clear at this point whether or not an RPCSEC_GSS-compliant null-byte or evasion message could be created. It is also not immediately clear what advantages RPCSEC_GSS holds over non-AUTH_NULL RPC authentication mechanisms in terms of this type of attack.

  These are obvious workarounds, but they don't really go to the core of the problem. A better solution would be to compensate for null-byte messages in the RPC protocol itself through a bugfix or algorithm redesign. This could be accomplished in at least two ways. One would be to detect and kill the null-byte message immediately upon receipt and shut down the offending session. This introduces another hidden denial-of-

service possibility if a small null-byte message is sent expressly for the purpose of making portmap/rpcbind close down communications with an otherwise legitimate client. The second method would also detect null-byte, but instead of shutting down the connection, it would instead write a report of the event to syslog. From there, a host-based IDS could pick up the syslogged notification and report to its server or management console. This solution adds case-specific IDS functionality to the RPC protocol.

We have not found a legitimate need for null-byte messages in the normal course of RPC traffic. We have also not found a case in which the "Last Fragment" portion of an inbound RPC message should be set to "No", either, even if it has a valid "Fragment Length" and associated data bytes properly encoded. Both of these conditions could be detected in the RPC protocol stack and reported to syslog when they occur as "Protocol Evasion" attempts. The "Protocol as NIDS" concept may be able scale out to encompass protocols other than RPC. This has the advantage of turning some portion of network core communication mechanisms into malicious activity detectors.

- **Observation:** IDS Evasion and DoS/DDoS are possible only in TCP. UDP is not at risk.

  portmap/rpcbind runs on TCP and UDP at ports 111 and 32771 through 32779. The reason UDP is not affected is because LF/FL messages are not used in UDP-based conversations. Data cannot be fragmented, so null-byte encoding is not possible in UDP. Encoding variations targeted at IDS evasion are also not possible in UDP-based messages.

- **Observation:** There are far too many valid RPC encodings for a signature-based IDS to handle.

  Solution: Detect portmap/rpcbind *replies* instead of requests. While there are practically an infinite number of possible *rpcinfo -p* encodings, there is only one reply – the portmap/rpcbind Dump Reply message discussed earlier. That reply will be consistently detectable by an IDS, unless the portmap/rpcbind host server has been compromised – in that case, there are much larger security problems to deal with than just protocol evasions. The IDS may not know what evasion method was used, but it will know a reply was sent. If the reply does not correlate with a normal *rpcinfo –p* request, then an evasion must have been used.

- **Observation:** IDSes that employ strict protocol decoding procedures open themselves up to the potential of resource-starvation and denial-of-service when processing null-byte RPC messages.

  Solution: An RPC-specific protocol decoding algorithm should be used that takes malicious encodings into account. The basic algorithm is as follows:

  Assumptions: Inbound data is directed at the portmap/rpcbind program (TCP ports 111 and 32771-32779)
  IP and TCP Headers are processed/stripped elsewhere

```
ALLOCATE BUFFER of variable length
LOOP
   READ LAST FRAGMENT/FRAGMENT LENGTH
      CASE
         LAST FRAGMENT === NO and FRAGMENT LENGTH ==
         Zero
            FLAG Level Two RPC Protocol Evasion
            STOP Processing this packet
            BREAK OUT OF CASE and LOOP
         LAST FRAGMENT != YES AND FRAGMENT LENGTH !=
         Zero
            FLAG Level One RPC Protocol Evasion
            READ number of bytes designated by FRAGMENT LENGTH
            into BUFFER
         LAST FRAGMENT == YES
            IF FRAGMENT LENGTH == Zero THEN
               FLAG Level One RPC Protocol Evasion
               SEND BUFFER to signature matching engine
               BREAK OUT OF CASE and LOOP
            ELSE READ number of bytes designated by FRAGMENT
            LENGTH into BUFFER
            SEND BUFFER to signature matching engine
            BREAK OUT OF CASE and LOOP
      END CASE
END LOOP
FREE BUFFER
```

Here is the same algorithm with comments:

```
ALLOCATE BUFFER of variable length
LOOP

# RPC calls are encoded by XDR which operates on data
# in chunks of length "n mod 4" (4, 8, 12, 16...bytes). The
# command strings are all four bytes, though.

    READ LAST FRAGMENT/FRAGMENT LENGTH
        CASE

        #
        # If the four bytes are all zero, it means a potential
        # resource-starvation/denial-of-service attack. Just flag the
        protocol
        # violation and let the outbound signatures handle it.
        # Put simply, this is a NO-OP attack of potentially infinite
        # length and we don't want to go there. ;)
        #

            LAST FRAGMENT == NO and FRAGMENT LENGTH ==
            Zero
                FLAG Level Two RPC Protocol Evasion
                STOP Processing this packet
                BREAK OUT OF CASE and LOOP

            #
            # This is the Sidestep evasion or a variation of it.
            # This evasion can be embedded in a NO-OP attack or it
            # can stand on its own.
            # If we detect Sidestep followed by NO-OP, we break
            # out of it in the NO-OP section above to prevent DoS.
            #

            LAST FRAGMENT != YES AND FRAGMENT LENGTH !=
            Zero
                FLAG Level One RPC Protocol Evasion
```

```
    READ number of bytes designated by FRAGMENT LENGTH into
    BUFFER

#
# "80 00" in the first two bytes means we are at the
# last fragment of data so we should prepare to wrap
# things up...
#

LAST FRAGMENT == YES

    #
    # A Sidestep evasion can be constructed that would
    # encapsulate all of the valid data before the "last
    # fragment" flag is set. In that case, the fragment
    # length must be zero - we need to be aware of that.
    #

    IF FRAGMENT LENGTH == Zero THEN
        FLAG Level One RPC Protocol Evasion

        #
        # "80 00 00 0x" means the BUFFER now has the
        # last fragment of data (and that data is of
        # length "x" bytes). If "x" is zero, then there's
        # no need to write anything to the output
        # BUFFER and we're done.
        #

        SEND BUFFER to signature matching engine
        BREAK OUT OF CASE and LOOP

    #
    # Otherwise, go ahead and fill out the rest of the
    # output BUFFER and send it to the signature engine.
    #

    ELSE READ number of bytes designated by FRAGMENT
    LENGTH into BUFFER
```

```
                SEND BUFFER to signature matching engine
                BREAK OUT OF CASE and LOOP
        END CASE
END LOOP
FREE BUFFER
```

The Bottom Line

At most, I'd assess the risk of the vulnerabilities described in this paper as medium. There are too many good existing workarounds to warrant anything more alarmist. Block TCP and UDP ports 111 and 32771 through 32779 at the head-end and intranet-to-intranet, deploy Wietse's portmap/rpcbind replacement and you'll be in pretty good shape at the local level. That doesn't help the long-haul networks much, but there's (arguably) a lot of junk on the wire already masquerading as valid content. ;-) This attack is just another brick in the DoS/DDoS wall from the perspective of practical network security.

For IDSes that employ strict protocol decoding, dealing with RPC protocol evasion techniques will be problematic. Null-byte streams could lead to resource starvation within the IDS itself. If a strict decode approach is pursued, we would expect the IDS to wait for a corresponding reply to the inbound query – and as we have demonstrated, that wait could be a long one. If numerous null-byte streams are sent simultaneously, the affected IDS will have numerous threads tied up in the decode process, hence the potential for resource starvation. Our results indicate that a modified protocol decode approach should be taken to avoid resource consumption. In essence, an IDS RPC decoder needs only to identify evasive fragmentation encoding techniques, flag that evasion attempt, and let its signature engine detect outbound replies to encoded queries. This modified decoding approach solves the resource-starvation problem, identifies evasion attempts, and detects query replies.

"…as through a glass, and darkly…"

Just some unanswered questions and other related thoughts:

- There is a lot of work yet to be done analyzing the RPC protocol, and I doubt I'll have a chance to get to it any time soon. For instance, while we know that initial RPC call setups must run though portmap/rpcbind, we don't know if the subsequent conversations that ensue after the handoff to RPC services such as mountd and the like can be evaded, DoS'ed, etc. Chances are good that there's probably more than a few bogeys waiting to be found down there.

- Are there really valid states in which Last Fragment can be set to No and carry data payloads? We haven't found any yet, but that doesn't mean they aren't supposed to be there. In the case of RPC calls through portmap/rpcbind, it would seem to me that fragmentation should be handled by IP – I don't see an immediate reason for allowing fragmentation inside RPC itself. Null-byte encoding is a Real Bad Thing, obviously. If UDP RPC doesn't need fragmentation support, why is it in TCP RPC?

- Can *any* protocol be evaded? That's where I'm going next. If there are unintentional consequences in RPC, you can bet there'll be some fun surprises elsewhere.

- Can the concept of "Protocol as NIDS" be extended beyond RPC? I'd like to think so because it puts intrusion detection capability into the core elements of the network itself…and that can only be a Real Good Thing.

Appendix A: Citation of Sources / List of References

1. Robert Graham

   SideStep: IDS evasion tool
   February 2001
   http://www.robertgraham.com/tmp/sidestep.html

2. RFC 1057

   RPC: Remote Procedure Call Protocol Specification Version 2
   June 1988 Sun Microsystems, Inc.
   http://www.rfc-editor.org/rfc/rfc1057.txt

3. RFC 1831

   RPC: Remote Procedure Call Protocol Specification Version 2
   R. Srinivasan August 1995 Sun Microsystems, Inc.
   http://www.rfc-editor.org/rfc/rfc1831.txt

4. RFC 1832

   XDR: External Data Representation Standard
   R. Srinivasan August 1995 Sun Microsystems, Inc.
   http://www.rfc-editor.org/rfc/rfc1832.txt

5. RFC 1833

   Binding Protocols for ONC RPC Version 2
   R. Srinivasan August 1995 Sun Microsystems, Inc.
   http://www.rfc-editor.org/rfc/rfc1833.txt

6. RFC 2203

RPCSEC_GSS Protocol Specification
M. Eisler, A. Chiu, L. Ling September 1997
http://www.rfc-editor.org/rfc/rfc2203.txt

7. Power Programming with RPC
John Bloomer
O'Reilly & Associates, 1st Edition February 1992
ISBN: 0-937175-77-3

8. OpenBSD Source Code
Various Distribution Revisions
http://www.openbsd.org/

9. FreeBSD Source Code
Various Distribution Revisions
The FreeBSD Project
http://www.freebsd.org

Appendix B: portmapper / rpcbind Protocol Decodes

Overview

The specific evasion technique *Sidestep* uses against the RPC protocol targets the portmap/rpcbind program. They are equivalent programs for the purposes of this discussion – portmap is used in the Linux and BSD variants of UNIX, rpcbind is used in the Solaris variant of UNIX. These programs are servers that convert incoming RPC program number queries into outgoing DARPA protocol port numbers. The evasion technique cloaks a portmap/rpcbind dump call in a manner which defeats standard IDS signature-matching, but still returns to the remote user a list of all registered RPC programs on the queried host. Dumps are an information-gathering technique used as a precursor for more advanced attacks against other RPC-based programs.

The basic form of a dump request is a TCP-based 40-byte data block encoded by the External Data Representation (XDR) module into ten 4-byte segments. This data is sent in one block from the remote system to the local host running portmap or rpcbind. The data block is decoded by XDR before being sent to portmap/rpcbind. The format of a dump call, with basic explanations of each field, is as follows. All data bytes are in hexadecimal representation:

Last Fragment/Fragment Length:	80 00 00 28 (Variable)
XID:	00 00 00 00 (Variable)
Message Type:	00 00 00 00 (Call)
RPC Version:	00 00 00 02 (Version = 2)
Program:	00 01 86 a0 (Portmap = 100000 decimal)
Program Version:	00 00 00 02 (Version = 2)
Procedure:	00 00 00 04 (Procedure = Dump)
Credential Flavor:	00 00 00 00 (Flavor = AUTH_NULL)
Credential Length:	00 00 00 00 (Length = 0)

Verifier Flavor:	00 00 00 00 (Flavor = AUTH_NULL)
Verifier Length:	00 00 00 00 (Length = 0)

The basic form of a portmapper dump reply is as follows:

Last Fragment/Fragment Length:	80 00 03 dc (Variable)
XID:	00 00 00 00 (Should correspond to originating request XID)
Message Type:	00 00 00 01 (Reply)
Reply State:	00 00 00 00 (Accepted)
Verifier Flavor:	00 00 00 00 (Flavor = AUTH_NULL)
Verifier Length:	00 00 00 00 (Length = 0)
Accept State:	00 00 00 00 (RPC executed successfully)
Value Follows:	00 00 00 01 (Yes)
Program:	00 01 86 a0 (Portmap = 100000 decimal)
Version:	00 00 00 04 (Version = 4)
Protocol:	00 00 00 06 (Protocol = TCP)
Port:	00 00 00 6f (Port = 111 decimal)
Value Follows:	00 00 00 01 (Yes)
.	
.	**Variable Length Records Follow**
.	
Value Follows:	00 00 00 00 (No)

Dump Reply Completed

Portmapper Version 2 Dump Call Breakdown

The byte fields in a portmap/rpcbind dump call are defined as follows. Explanatory information is provided where necessary.

Last Fragment:	Two known states: **Yes = 80 00** **No = 00 00**	XDR Encoding:
Fragment Length:	Variable: **Zero to ff ff**	
LF/FL Combined:		**80 00 00 00** . . **80 00 ff ff** . . **00 00 00 00** . . **00 00 ff ff**
XID:	Multiple potential states: **Zero to 2^32**	XDR Encoding: **00 00 00 00** … **ff ff ff ff**

The XID field is used to associate inbound calls with outbound replies. It is 32-bit number that can vary significantly from one call to the next.

Message Type:	Two defined states: **Call = 00** **Reply = 01**	XDR Encoding: **00 00 00 00** **00 00 00 01**

Note: The "reply" state is defined, but it should not Appear in a dump call – it should only appear in a dump reply.

RPC Version:	One defined state:		XDR Encoding:
	RPC Version = 2		**00 00 00 02**

Note: The only valid defined state for RPC Version is 2 at the present time.

Program:	At least 63 defined states:		XDR Encoding
	portmapper	**100000**	**00 01 86 a0**
	rstatd	**100001**	**00 01 86 a1**
	rusersd	**100002**	**00 01 86 a2**
	nfs	**100003**	**00 01 86 a3**
	ypserv	**100004**	**00 01 86 a4**
	mountd	**100005**	**00 01 86 a5**
	ypbind	**100007**	**00 01 86 a7**
	walld	**100008**	**00 01 86 a8**
	yppasswdd	**100009**	**00 01 86 a9**
	etherstatd	**100010**	**00 01 86 aa**
	rquotad	**100011**	**00 01 86 ab**
	sprayd	**100012**	**00 01 86 ac**
	3270_mapper	**100013**	**00 01 86 ad**
	rje_mapper	**100014**	**00 01 86 ae**
	selection_svc	**100015**	**00 01 86 af**
	database_svc	**100016**	**00 01 86 b0**
	rexd	**100017**	**00 01 86 b1**
	alis	**100018**	**00 01 86 b2**
	sched	**100019**	**00 01 86 b3**
	llockmgr	**100020**	**00 01 86 b4**
	nlockmgr	**100021**	**00 01 86 b5**
	x25.inr	**100022**	**00 01 86 b6**
	statmon	**100023**	**00 01 86 b7**
	status	**100024**	**00 01 86 b8**
	bootparam	**100026**	**00 01 86 b9**
	ypupdated	**100028**	**00 01 86 ba**
	keyserv	**100029**	**00 01 86 bb**
	sunlink_mapper	**100033**	**00 01 86 c1**
	tfsd	**100037**	**00 01 86 c5**
	nsed	**100038**	**00 01 86 c6**

nsemntd = 100039 00 01 86 c7

showfhd	=	100043	00 01 86 cb
ioadmd	=	100055	00 01 86 d7
NETlicense	=	100062	00 01 86 de
sunisamd	=	100065	00 01 86 e1
debug_svc	=	100066	00 01 86 e2
ypxfrd	=	100069	00 01 86 e5
bugtraqd	=	100071	00 01 86 e7
kerbd	=	100078	00 01 86 ee
event	=	100101	00 01 87 05
logger	=	100102	00 01 87 06
sync	=	100104	00 01 87 08
hostperf	=	100107	00 01 87 0b
activity	=	100109	00 01 87 0d
hostmem	=	100112	00 01 87 10
sample	=	100113	00 01 87 11
x25	=	100114	00 01 87 12
ping	=	100115	00 01 87 13
rpcnfs	=	100116	00 01 87 14
hostif	=	100117	00 01 87 15
etherif	=	100118	00 01 87 16
iproutes	=	100120	00 01 87 18
layers	=	100121	00 01 87 19
snmp	=	100122	00 01 87 1a
traffic	=	100123	00 01 87 1b
nfs_acl	=	100227	00 01 87 83
sadmind	=	100232	00 01 87 88
ufsd	=	100233	00 01 87 89
nisd	=	100300	00 01 87 cc
nispasswd	=	100303	00 01 87 cf
pcnfsd	=	150001	00 02 49 f1
amd	=	300019	00 04 93 f3
bwnfsd	=	545580417	20 84 e5 81
fypxfrd	=	600100069	23 c4 cc e5

Program Version:	Multiple potential states:		XDR Encoding:
	Zero to 2^32		**00 00 00 00** … **ff ff ff ff**
	Note: In practice, this value can be expected to be less than 10 decimal.		**00 00 00 00** … **00 00 00 0a**
Procedure:	Multiple defined states:		XDR Encoding:
	NULL	**= 0**	**00 00 00 00**
	SET	**= 1**	**00 00 00 01**
	UNSET	**= 2**	**00 00 00 02**
	GETADDR	**= 3**	**00 00 00 03**
	DUMP	**= 4**	**00 00 00 04**
	CALLIT	**= 5**	**00 00 00 05**
	GETTIME	**= 6**	**00 00 00 06**
	UADDR2TADDR	**= 7**	**00 00 00 07**
	TADDR2UADDR	**= 8**	**00 00 00 08**
Credentials:			
Flavor:	Multiple defined states:		XDR Encoding:
	AUTH_NONE	**= 0**	**00 00 00 00**
	AUTH_NULL	**= 0**	**00 00 00 00**
	AUTH_SYS	**= 1**	**00 00 00 01**
	AUTH_UNIX	**= 1**	**00 00 00 01**
	AUTH_SHORT	**= 2**	**00 00 00 02**
	AUTH_DES	**= 3**	**00 00 00 03**
	AUTH_DH	**= 3**	**00 00 00 03**
	AUTH_KERB	**= 4**	**00 00 00 04**
	RPCSEC_GSS	**= 6**	**00 00 00 06**

Note: RPCSEC_GSS authentication is defined in RFC 2203 "RPCSEC_GSS Protocol Specification" but it is not implemented in current portmapper/rpcbind source code.

Length:	Multiple potential states:	XDR Encoding:
	For AUTH_NULL or AUTH_NONE, Length will be zero (0).	**00 00 00 00**

Verifier:

Flavor:	Multiple defined states:	XDR Encoding:
	AUTH_NONE = 0	**00 00 00 00**
	AUTH_NULL = 0	**00 00 00 00**
	AUTH_SYS = 1	**00 00 00 01**
	AUTH_UNIX = 1	**00 00 00 01**
	AUTH_SHORT = 2	**00 00 00 02**
	AUTH_DES = 3	**00 00 00 03**
	AUTH_DH = 3	**00 00 00 03**
	AUTH_KERB = 4	**00 00 00 04**

Length:	Multiple potential states:	XDR Encoding:
	For AUTH_NULL or AUTH_NONE, Length will be zero (0).	**00 00 00 00**

Portmapper Version 2 Dump Reply Breakdown

The byte fields in a portmapper dump reply are defined in two sections (RPC Call Portion and Portmapper Reply Portion) as follows:

RPC Call Portion

XID:	Multiple potential states:	XDR Encoding:
	Zero to 2^32	**00 00 00 00**
		…
		FF FF FF FF

Note: The reply XID should correspond to the call XID

The XID field is used to associate inbound calls with outbound replies. It is a 32-bit number that can vary significantly from one call to the next.

Message Type:	Two defined states:		XDR Encoding:
	Call	= **00**	**00 00 00 00**
	Reply	= **01**	**00 00 00 01**

Note: The "reply" state is defined, but it should not Appear in a dump call – it should only appear in a dump reply.

Reply State:	Two defined states:		XDR Encoding:
	Accepted	= **00**	**00 00 00 00**
	Denied	= **01**	**00 00 00 01**

Verifier:

Flavor:	Multiple defined states:		XDR Encoding:
	AUTH_NONE	= **0**	**00 00 00 00**
	AUTH_NULL	= **0**	**00 00 00 00**
	AUTH_SYS	= **1**	**00 00 00 01**
	AUTH_UNIX	= **1**	**00 00 00 01**
	AUTH_SHORT	= **2**	**00 00 00 02**
	AUTH_DES	= **3**	**00 00 00 03**
	AUTH_DH	= **3**	**00 00 00 03**
	AUTH_KERB	= **4**	**00 00 00 04**

Length:	Multiple potential states:	XDR Encoding:
	For AUTH_NULL or AUTH_NONE, Length will be zero (0).	**00 00 00 00**

Accept State:	Six defined states:		XDR Encoding:
	Success	= **00**	**00 00 00 00**
	Program Unavailable	= **01**	**00 00 00 01**
	Program Mismatch	= **02**	**00 00 00 02**
	Procedure Unavailable	= **03**	**00 00 00 03**
	Garbage Args	= **04**	**00 00 00 04**

System Errror = **05 00 00 00 05**

Portmapper Reply Portion

Value Follows:	Two defined states:	XDR Encoding:
	No = **00**	**00 00 00 00**
	Yes = **01**	**00 00 00 01**

Program:	At least 63 defined states:	XDR Encoding
	Portmapper = **100000**	**00 01 86 a0**
	rstatd = **100001**	**00 01 86 a1**
	rusersd = **100002**	**00 01 86 a2**
	nfs = **100003**	**00 01 86 a3**
	ypserv = **100004**	**00 01 86 a4**
	mountd = **100005**	**00 01 86 a5**
	ypbind = **100007**	**00 01 86 a7**
	walld = **100008**	**00 01 86 a8**
	yppasswdd = **100009**	**00 01 86 a9**
	etherstatd = **100010**	**00 01 86 aa**
	rquotad = **100011**	**00 01 86 ab**
	sprayd = **100012**	**00 01 86 ac**
	3270_mapper = **100013**	**00 01 86 ad**
	rje_mapper = **100014**	**00 01 86 ae**
	selection_svc = **100015**	**00 01 86 af**
	database_svc = **100016**	**00 01 86 b0**
	rexd = **100017**	**00 01 86 b1**
	alis = **100018**	**00 01 86 b2**
	sched = **100019**	**00 01 86 b3**
	llockmgr = **100020**	**00 01 86 b4**
	nlockmgr = **100021**	**00 01 86 b5**
	x25.inr = **100022**	**00 01 86 b6**
	statmon = **100023**	**00 01 86 b7**
	status = **100024**	**00 01 86 b8**
	bootparam = **100026**	**00 01 86 b9**
	ypupdated = **100028**	**00 01 86 ba**
	keyserv = **100029**	**00 01 86 bb**
	sunlink_mapper = **100033**	**00 01 86 c1**
	tfsd = **100037**	**00 01 86 c5**

nsed	=	100038	00 01 86 c6
nsemntd	=	100039	00 01 86 c7
showfhd	=	100043	00 01 86 cb
ioadmd	=	100055	00 01 86 d7
NETlicense	=	100062	00 01 86 de
sunisamd	=	100065	00 01 86 e1
debug_svc	=	100066	00 01 86 e2
ypxfrd	=	100069	00 01 86 e5
bugtraqd	=	100071	00 01 86 e7
kerbd	=	100078	00 01 86 ee
event	=	100101	00 01 87 05
logger	=	100102	00 01 87 06
sync	=	100104	00 01 87 08
hostperf	=	100107	00 01 87 0b
activity	=	100109	00 01 87 0d
hostmem	=	100112	00 01 87 10
sample	=	100113	00 01 87 11
x25	=	100114	00 01 87 12
ping	=	100115	00 01 87 13
rpcnfs	=	100116	00 01 87 14
hostif	=	100117	00 01 87 15
etherif	=	100118	00 01 87 16
iproutes	=	100120	00 01 87 18
layers	=	100121	00 01 87 19
snmp	=	100122	00 01 87 1a
traffic	=	100123	00 01 87 1b
nfs_acl	=	100227	00 01 87 83
sadmind	=	100232	00 01 87 88
ufsd	=	100233	00 01 87 89
nisd	=	100300	00 01 87 cc
nispasswd	=	100303	00 01 87 cf
pcnfsd	=	150001	00 02 49 f1
amd	=	300019	00 04 93 f3
bwnfsd	=	545580417	20 84 e5 81
fypxfrd	=	600100069	23 c4 cc e5

Program Version:	Multiple potential states: **Zero to 2^32**	XDR Encoding: 00 00 00 00 … ff ff ff ff
	Note: In practice, this value can be expected to be less than 10 decimal.	00 00 00 00 … 00 00 00 0a

Protocol:	Four defined states:		XDR Encoding:
	No Protocol	= 00	00 00 00 00
	ICMP	= 01	00 00 00 01
	TCP	= 06	00 00 00 06
	UDP	= 17	00 00 00 11

Port:	Multiple potential states:		XDR Encoding:
	Port	= 111	00 00 00 6f

Note: The value returned in the "Port" variable can be any legitimate TCP or UDP port number.

Value Follows:	Two defined states:		XDR Encoding:
	No	= 00	00 00 00 00
	Yes	= 01	00 00 00 01

Note: The "Value Follows" Block defines the start and end of unstructured data. "Value Follows" will be Yes until all data has been sent. The last 4-byte string sent will be the Value Follows block set to "No" (00 00 00 00 in XDR encode).

Appendix C: portmap / rpcbind IDS Evasion Techniques

Sidestep IDS Evasion

The *Sidestep* program written by Robert Graham uses the following 200-byte data packet when cloaking a portmap/rpcbind dump request. The data is presented in 32-byte segments, left-to-right, top-to-bottom:

```
00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00     00 00 01 00 00 00
00 01 00 00 00 00 01 00 00 00

00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00     01 00 00 00 00 01
00 00 00 00 01 02 00 00 00 01

00 00 00 00 01 01 00 00 00 01 86 00 00 00 01 A0     00 00 00 01 00 00
00 00 01 00 00 00 00 01 00 00

00 00 01 02 00 00 00 01 00 00 00 00 01 00 00 00     00 01 00 00 00 00
01 04 00 00 00 01 00 00 00 00

01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01     00 00 00 00 01 00
00 00 00 01 00 00 00 00 01 00

00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00     00 00 01 00 00 00
00 01 00 00 00 00 01 00 00 00

00 01 00 80 00 00 01 00
```

The data stream above appears to be significantly different from a standard portmap/rpcbind dump request data stream and successfully evades detection by signature-based IDSes…

```
80 00 00 28 00 00 00 00 00 00 00 00 00 00 00 02     00 01 86 A0 00 00
00 02 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

…but portmap/rpcbind treats both data streams identically as valid dump requests. The questions are, obviously, "How does the evasion work?", "What are its implications?", and "How can signature-based IDSes be improved to detect any RPC evasion?"

How *Sidestep* Works

The key to understanding how *Sidestep* works is the key to understanding how the entire portmap/rpcbind state machine works. The key is: **portmap/rpcbind will not process any inbound information requests until it knows that it has received all of the data that is going to be sent by the requestor.**

The "Last Fragment/Fragment Length" header that begins the RPC data payload tells portmap/rpcbind whether or not it has received the last fragment of data, and how many bytes of data are contained in the payload. Let's look again at a raw normal portmap/rpcinfo dump request:

80 00 00 28 00 00 00 00 00 00 00 00 00 00 00 02 00 01 86 A0 00 00
00 02 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Now let's break this raw data into chunks that have meaning in terms of portmap/rpcbind:

80 00 00 28 Explanation: **80 00** means this is the last fragment of data.

00 28 means the data fragment is 40 bytes (28 hexadecimal) in length.

00 00 00 00

00 00 00 00 Indeed, there are 40 more bytes of data to be read

00 00 00 02 in. The underlined fields are the byte combinations

00 01 86 a0 most signature-based IDSes trigger on to warn of an

00 00 00 02 incoming portmap/rpcbind dump request. See

00 00 00 04 Appendix B for what these data fields represent.

00 00 00 00

00 00 00 00 This represents data encoding. *Sidestep*

00 00 00 00 evasion is really nothing more than a valid, albeit

00 00 00 00 uncommon, encoding technique.

Sidestep Encoding Technique

Let's take another look at the 200-byte data packet *Sidestep* sends to portmap/rpcbind and manually decode it.:

00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00

00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 01 02 00 00 00 01

00 00 00 00 01 01 00 00 00 01 86 00 00 00 01 A0 00 00 00 01 00 00 00 00 01 00 00

00 00 01 02 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 04 00 00 00 01 00 00 00 00

01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 01 00 00 00 00 01 00

00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00

00 01 00 80 00 00 01 00

All bytes in the data packet except for the last five decode as follows. Encoding instructions are in bold. Data bytes are underlined:

00 00 00 01 <u>00</u>

Translating this to English is straightforward:

00 00 00 01 <u>00</u> **Last Fragment? (No = 00 00)**

Fragment Length? (One byte = 00 01)

Data? <u>00</u>

Applying that encoding technique to the entire data packet, we get this:

00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 02 00 00 00 01 00 00 00 00 01 01 00 00 00 01 86 00 00 00 01 A0 00 00 00 01 00 00 00 00 01 00 00 00 01 02 00 00 00 01 00 00 00 00 01 00 00 00 01 04 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 00 00 00 01 00 80 00 00 01 00

If we align the data on four-byte boundaries, the encoding is easier to understand:

Encoded Data Stream **Decoded Data Stream**

00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *0000 00 00 00*
(XID)

00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *0000 00 00 00*
(Msg Type)

00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *0200 00 00 00*
(RPC Version)

00 00 00 01 *00* 00 00 00 01 *01* 00 00 00 01 *86* 00 00 00 01 *A0 00 01 86 a0*
(Program)

00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *0200 00 00 02*
(Pgm Version)

00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *0400 00 00 04*
(Procedure)

00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *0000 00 00 00*
(Cred Flavor)

00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *0000 00 00 00*
(Cred Length)

00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *0000 00 00 00*
(Auth Flavor)

00 00 00 01 *00* 00 00 00 01 *00* 00 00 00 01 *00* 80 00 00 01 *0000 00 00 00*
(Auth Length)

The column of data on the right looks exactly like a portmap/rpcbind dump request because that is exactly what it is. The last five bytes are underlined to illustrate an important point about portmap/rpcbind. Let's decode just those bytes:

80 00 00 01 *00* **Last Fragment? (Yes = 80 00)**

Fragment Length? (One byte = 00 01)

Data? *00*

These bytes tell portmap/rpcbind that it now has all of the data and what that last byte of data is. It is important to understand that portmap/rpcbind will not actually process *anything* until it knows it has the last data fragment. This feature makes *Sidestep* evasion possible. But the encoding feature *Sidestep* uses as a static evasion technique is not the only one possible.

Beyond *Sidestep*

The encoding technique used by *Sidestep* is just one of many potential encodings, and hence possible evasions that will work against signature-based IDSes. The number of possible valid encoding combinations for a portmap/rpcbind dump request is 40! or approximately 8.159×10^{47}. This is the upper limit because only 40 bytes of data are going to be encoded. Now that we know the technique, we can create encoding combinations manually. Let's create a more complex encoding:

Rules:

1. All fragment message byte values used must add up to 40.
2. All data bytes must be encoded properly in the message.
3. All data fragment messages except the last must indicate that more data fragments follow and must properly indicate the number of data bytes that follow.
4. A single last fragment message must be sent with valid length and associated data bytes.

Let's use the following encoding values:

7, 3, 5, 9, 2, 6, 7, 1

The data payload would then look like this:

00 00 00 07 ***00 00 00 00 00 00 00*** **00 00 00 03** ***00 00 00*** **00 00 00 05** ***00 02 00 01 86***

00 00 00 09 ***a0 00 00 00 02 00 00 00 04*** **00 00 00 02** ***00 00*** **00 00 00 06** ***00 00 00 00***

00 00 **00 00 00 07** ***00 00 00 00 00 00 00*** **80 00 00 01** ***00***

It should be fairly simple to construct an RPC protocol decoder from the information provided as a part of a signature-based IDS which would reconstruct the encoded data packet and hand the results to the signature-matching engine. Unfortunately, however, there is one valid encoding case that makes decoding not only difficult, but deadly.

Null-Byte Encoding

LF/FL fragmentation messages control the course of communications with portmap/rpcbind. So far, we have dealt with their relatively sane states, **80 xx** or **00 xx**, where **xx** had some non-zero value. **80** for the Last Fragment message (80 for Last Fragment = Yes) with a zero or non-zero FL is a message terminator, but **00** (Last Fragment = No) with a zero FL can extend the duration of a message to portmap/rpcbind to near infinity. Consider the implications of this encoding:

00 00 00 00 …

This encoding means we do not have the last data fragment (LF) and the fragment length (FL) is also zero. portmap/rpcbind cannot process and data until it has been told the last data fragment has been transmitted so this case forces portmap/rpcbind into a loop. Because the conversation is taking place over TCP, it can go on forever. This encoding capability appears to be a flaw in the original design of RPC communication algorithms that has persisted from its creation until the present time. We have demonstrated in our lab environment that portmap/rpcbind can be locked using this encoding in a loop of hundreds of millions of such messages. It will not respond to any other external input – in other words, it becomes a denial-of-service attack. If this encoding is applied to tools such as Stacheldraht, it scales dramatically to a DDoS attack. We have not found that portmap/rpcbind will crash or overflow in any way that would yield a root shell as a result of this attack, but we have not pushed very hard against the boundaries of sanity. Null-byte encoding also allows the embedding of legitimate requests and evasion attempts within it if the data bytes are encoded correctly. Without null-byte, the number of possible legitimate encodings is 40! – with null-byte thrown into the mix, for practical purposes that number goes to infinity.

IDSes that perform protocol decoding are also susceptible to this attack if they adhere to a strict interpretation of RPC communication rules. They too will wait forever for that all-important Last Fragment message. If the IDS in question is monitoring network head-end or some high-traffic point and a null-byte RPC encoding is amplified by a tool such as Stacheldraht, it is likely the IDS will go into resource-starvation. This is not a good condition for either the network or the IDS monitoring it.

Appendix D: Acknowledgements / Bio

"I'd like to thank everyone on behalf of the group and ourselves and I hope we
passed the audition."

--John Lennon---

My wife, Terri, who taught me that while all things are certainly possible, all things are not necessarily good things so keep yer damned eyes open… The entire staff of Network Security Wizards, now the Enterasys Dragon Team. Leevi Marttila's C to English and English to C translator for the Blowfish algorithm, which was the inspiration for the "In Plain English" discussion of RPC. George Patton and Ethereal. Louis, Natasha, Isis, and Starbuck the Wonder Rabbit. The Lerxst and Pong-Fu. Soundtrack by Foo Fighters, The Beatles, Radiohead, and Pink Floyd. And Pete Townsend would like to thank the Who…again.

Randy Taylor (the "Joseph R." is just for official stuff) and his wife pay rent to two Siberian Huskies, a cat, and a rabbit for their home in the concrete and asphalt tundra that is Suburban Maryland. He has worked in computer and network security since 1988. His sense of humor is reported to be not just twisted, but utterly bent. He tends to say pretty much exactly what's on his mind at the time – some describe this as a character flaw, but Randy blames it on his exceptionally strong aversion to BS and his belief that communication is only possible among equals.

Assignment 2 – "Network Detects"

Intrusion Detection In Depth
GCIA Practical Assignment

Joseph Taylor
Version 2.9x

1. Source of Traces (ALL DETECTS):

All trace sources are from my home network.

2. Detect Generated By (ALL DETECTS):

All detects were generated by the Dragon Intrusion Detection System.

Let's look at the five detects, listed below in the Dragon Fire event correlation output:

sum_event

Sep 19, 2001

EVENT NAME	COUNT	LAST TIME	LAST SOURCE IP		LAST DEST IP	TIME CHART
[SAFEWEB]	20	22:35:59	Not.My.Net.10	= >	My.Net.Wrk.34	.....++.
[ZKS:FREEDOM-SETUP]	2	22:19:27	Not.My.Net.202	= >	My.Net.Wrk.34	.....+.....+.
[GAMES:ANARCHY-ONLINE-1]	4	22:18:28	Not.My.Net.11	= >	My.Net.Wrk.34	.....++.
[SSH:VERSION-1]	1	15:19:54	My.Net.Wrk.223	= >	My.Net.Wrk.101	.....+.....

Sep 23, 2001

EVENT NAME	COUNT	LAST TIME	LAST SOURCE IP		LAST DEST IP	TIME CHART
[PGP-EMAIL]	16	14:40:43	My.Net.Wrk.34	= >	Not.My.Net.29	.....+.....

Safeweb

3. Probability the source address was spoofed:

In this case, the spoofing probability is zero because I generated this detect myself using Safeweb.
Safeweb requires the establishment of a TCP three-way handshake so the session source address cannot be spoofed. However, it can be chained onto other privacy technologies, such as Freedom, which can completely obfuscate the true source of the session.

4. Mechanism Description – What is *Safeweb?*:

From https://fugu.safeweb.com/sjws/solutions/safeweb.html

SAFEWEB

Our basic consumer technology lets you surf the Web anonymously so that no one can pry into your online communications. As long as you have access to the Internet, you can use SafeWeb **for free** by going to www.safeweb.com. Unlike other "free" privacy services, ours has no strings attached -- no time delays, no user quotas, no withholding of critical features such as encryption.

FEATURES

Free: no strings attached

Easy to use: Web-based privacy solution accessible to all Internet users anywhere, anytime — **no time-consuming or complicated downloads, installation or configuration**

Effective: 128-bit SSL encryption of all content and URLs, ensuring that all data exchanged between the user's computer and SafeWeb is completely private

Universally accessible: compatible with all major operating platforms and browsers (e.g., Netscape, Internet Explorer, Windows, Mac, Linux, Sun)

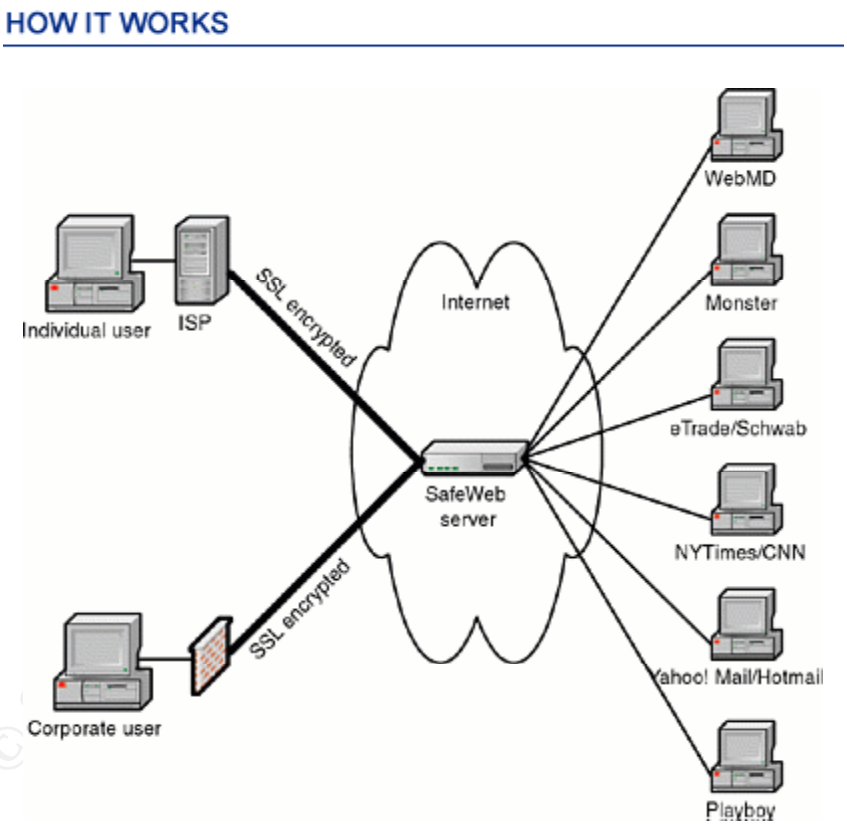
Functional: The only free Web-based privacy solution that effectively sanitizes and rewrites DHTML (e.g., JavaScript, VBScript, CSS) so that users may access popular rich Web sites such as Sony, MTV, Hotmail, Webvan, eTrade and countless others

User preferences: Users can control their level of privacy by setting security features to filter or block cookies, Web bugs/pop-up windows and IP data, and to disable potentially harmful JavaScript commands. SafeWeb also allows users to set bookmarks.

5. Mechanism Functionality – How Does *Safeweb* work?:

Safeweb establishes a 128-bit SSL session between the requesting web browser client and the Safeweb proxy server. Once established, all client requests for web-based functionality are encrypted on the client-side, reappearing in cleartext on the content-side of the Safeweb cloud. Servers on the content-side of the cloud believe the Safeweb cloud to be the originator of the session requests and respond normally. Safeweb then encrypts that content and returns it to the browser client.

From
https://fugu.safeweb.com/sjws/solutions/how_it_works_safeweb.html



The diagram clearly shows that once the SSL session is active between the requesting individual or corporate browser, traffic content (URL requests, etc.) will be encrypted. The user of the browser leverages the Safeweb network to securely and anonymously interact with any web-based resource, which would include anonymized email sources such as Yahoo and Hotmail.

However, Safeweb's implementation requires an X.509 certificate exchange

between the browser and the Safeweb server. That exchange happens in cleartext on TCP port 443 and can be detected by the Dragon IDS signature **[SAFEWEB]**.

The Dragon Detect of [SAFEWEB]

The Safeweb X.509 certificate exchange is captured via Dragon's session reconstruction interface. Safeweb communications to my network are in BLUE. My browser responds in RED. After this exchange, all further session traffic is completely encrypted. Detection triggers are highlighted.

Sep 19, 2001

{16}{3}{0}{0}J{2}{0}{0}F{3}{0};{A9}6;{D6}{3}6c{B1}{11}t{10}{C5}{CC}{E
9}s{8A}{86}7{8E}{9F}{0}{87}`:c{80}{C6}{3} < {C1}{9B}
U{CB}{A0}{E5}{E7}{3}{1C}{DA}:!0#{8D}{A4}%{B6}.{A6}{C9}{C3}{C4}{B}{B6}
{DF}{94}{CD}{CE}{B0}AOq{9A}{0}
{4}{0}{16}{3}{0}{2}{EF}{B}{0}{2}{EB}{0}{2}{E8}{0}{2}{E5}0{82}{2}{E1}0
{82}{2}J{A0}{3}{2}{1}{2}{2}{3}{7}
U{4}{8}{13}{C}Western Cape1{12}0{10}{6}{3}U{4}{7}{13}{9}Cape
Town1{1D}0{1B}{6}{3}U{4}{A}
{13}{14}Thawte Consulting cc1(0&{6}{3}U{4}{B}{13}{1F}Certification
Services Division1{19}0{17}{6}{3}U{4}{3}{13}{10}
Thawte Server CA1&0${6}{9}*{86}H{86}{F7}{D}{1}{9}{1}{16}{17}server-
certs@**thawte.com**0{1E}{17}{D}010409191330Z{17}{D}
020409191330Z0t1{B}0{9}{6}{3}U{4}{6}{13}{2}US1{13}0{11}{6}{3}U{4}{8}{
13}{A}California1{10}0{E}{6}{3}U{4}{7}{13}{7}Oakland1{16}0{14}{6}{3}U
{4}
{A}{13}{D}**Safeweb, Inc.**1{C}0{A}
{1}{1}{5}{0}{3}{81}{8D}{0}0{81}{89}{2}{81}{81}{0}{C6}oU2{83}{E3}{D8}{
EF}{1D}w{E6}{1}{A3}c{16}Pj{DB}{81}H{85}{FA},{A2}{99}{AF}{9C}{E1}{CA}{
D6}]{5}{A1}{B8}a{8A}{D1}{A7}x{90}{1B}D^{AA}{99}{FC}{1D}&{83}{1E}{99}{
89}{6}{D6}{84}M{CE}{B8}Y
{E7}{F5}2{81}{AB}{9C}1i{84}{13}{BD}:{A1}{A8}{D2}m{CD}{E3}{AB}{9C}{BC}
{F5}{80}{8A}^{F1}@{CC}{81}{D0}{B4}{91}Y{C9}{ED}"Q{9A}
{D1}/{13}a{BF}R{D}7{EC}B/{B4}Y{DF}{6}{F5}C{EC}{A}
{1E}{E3}{AD}On{1F}{A4}~{88}D7{B9}{2}{3}{1}{0}{1}{A3}00.0{1E}{6}{3}U{1
D}%{4}{17}0{15}{6}{8}+{6}{1}{5}{5}{7}
{1}{1}{4}{5}{0}{3}{81}{81}{0}N{BA}{F}{BB}{3}98%{EB}{1A}{F7}L{12}{86}{
AD}{86}{4}{{D9}{85}{9F}{8F}f{DC}{B7}gW{83}
{C}{E9}5{3}{F0}{8A}{BC}{1A}{E0}{C9}{C3}{F6}
{8D}{1E}{8D}{AF}F{DC}{A4}{DA}{2}{AC}R{7}z{AA}{C7}J3j{1A}{D3}{F8}{AE}{
11}$u{CD}{81}ozn{9C}{D0}{14}{AC}{1F}{90}gW({B5}[{C6}4{FB}{AD}{96}~J{E

```
}{86}{A4}{A2}O{3}lDk{89}g{EA}{11}{{B}u{1B}U{97}6{E9}q
{8D}{83}Z{8C}{C2}{95}{BD}q@{A6}{8F}{9C}O{DB}{A7}{95}p{16}{3}{0}{0}{4}
{E}{0}{0}{0}{A}

{16}{3}{0}{0}{84}{10}{0}{0}{80}'{3}{B3}- < {CC} >
{97}{9A}A{B4}{A8}{90}2{B3}{F}{A3}{AF}{B8}h{9B}{C0}{D2}{BB}
{5}Bim{B2}0{{9A}{D9}{7F}{91}{18}{1C}}{C0}E{3}]p}{E0}{83}#{D1}{AE}{FA}
{1A}6`{C9}{F4}{8A}k\&A{B9}]{87}{FD}{E6}{C8}C{B8}{A6}
{CF}v{C3}&{1E}{12}{90}{CD}{AA}[P{AC}{FF}VK,Zk{EF}{80}{E3}m{6}{DD}{BE}
e{A7}4{C0}{FB}{E5}a{B9}8{4}{F}b{E1}Q{C}GLK/{8B}{3}
{13}{C1}{11}{B3}{A6}E={2}{FE}{CE}v{7F}{C3}yXP{E}C[{1D}{BA}{2}{B1}{1A}
{DA}{8E}{EC}{F7}{8E}dz%{E0}x{1D}{E1}{F1}{7F}2{13}w{A}

{14}{3}{0}{0}{1}{1}{16}{3}{0}{0}8{13}{B2}{CC}{90}{E3}{A}
{EE}{F6}{DA}{9E}{B8}{DF}(+D{86}{CC}{A5}b{D3}{B1}{0}M{A}
{DE}T{AB}VpSZ,{E7}{EC}{17}{A5}{F4}M{A7}){1B}m{D6}Y+{10}m{DC}{7F}{F8}
{EA}a{B0}{92}{E3}{FB}
```

6. Correlations:

The only correlation I could find is a "content-side" Safeweb detect reported by John Benninghoff in the 11/10/2000 GIAC incident handler's log at http://www.incidents.org/archives/y2k/111000.htm John's detect appears to be related to Safeweb-internal operations instead of client-side browser operation. There is not enough information to assert that client-side requests correlate to the "Proximity-Checking" behavior reported by Mr. Benninghoff.

7. Evidence of Active Targeting:

The client-side web browser must connect at https://www.safeweb.com/ This constitutes Active Targeting in that use of Safeweb is a conscious act.

8. Severity:

All estimates of severity in the use of Safeweb are subjective because they are dependent on the network policies of the observer. For instance, a Safeweb session originating from a .mil domain is subjectively of more concern that one originating from a .org site. Therefore, the full min-max range of zero to five must be assumed.

Target Criticality(TC):	0 (If network policy allows) 3 (Use from .com domain) 5 (Use from .mil domain)
Use Lethality(UL):	0 (Mechanism is not a vulnerability in itself, use allowed) 5 (Use in violation of policy, suspicious activity)
System Countermeasures(SC):	0 (Use allowed – invoked via Web browser)
Network Countermeasures(NC):	0 (Use allowed, no blocking in place) 5 (Use not allowed, blocking in place)

Use Severity:

Case 1: Normal home use
(TC + UL) – (SC + NC) = (0 + 0) - (0 + 0) = 0

Case 2: .com use allowed
(TC + UL) – (SC + NC) = (3 + 0) - (0 + 0) = 3

Case 3: .com use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 0) = 8

Case 4: .com use violates policy, blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 5) = 3

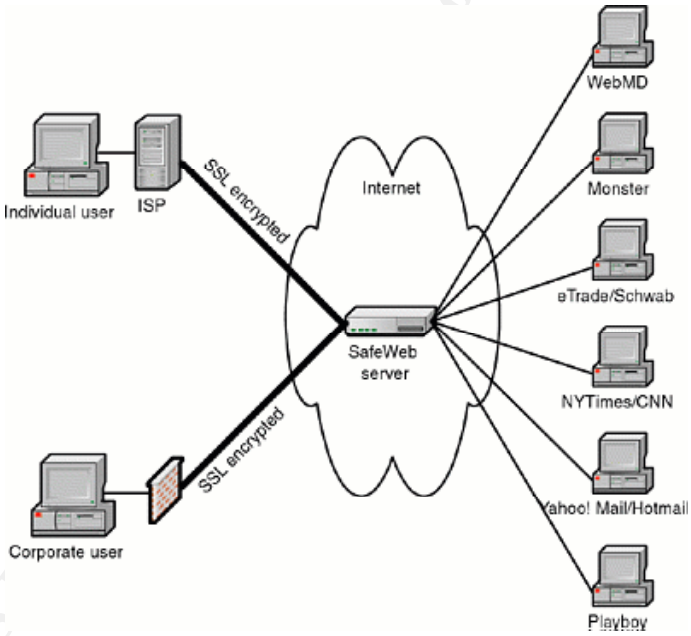
Case 5: .mil use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (5 + 5) - (0 + 0) = 10

9. Defensive Recommendation:

Evaluate the use of Safeweb against organization Acceptable Use Policies (AUP), if any. If Safeweb violates AUP, block all web connections to the safeweb.com domain.

10. Test Question:

Using the diagram below, where besides the client-side can Safeweb use be detected?



(a) There is no other detection point.
(b) Between the Safeweb cloud and its destination(s).
(c) Psst... Hey Guido. It's all so clear to me now.
(d) Inside the Safeweb cloud.

Answer: (b)

FREEDOM
PREMIUM SERVICES

Freedom

3. Probability the source address was spoofed:

In this case, the spoofing probability is zero because I generated this detect myself using Freedom.
Freedom requires the establishment of a TCP three-way handshake so the session source address cannot be spoofed. It cannot be chained onto other privacy technologies because it is a host-based application that uses its own "shim" in the network stack. However, some privacy products, such as Safeweb can be used in conjunction with Freedom.

4. Mechanism Description – What is *Freedom*?:

From:
http://www.freedom.net/products/premium/index.html?product=premium

Key Features & Benefits

Go online undetected and prevent websites from tracking your activities
Anonymous Web Browsing Service
Enjoy private, untraceable access to the Internet and erase the tracks you leave when you browse the Web, post to newsgroups or chat rooms, or telnet to other computers. With our Anonymous Browsing service, no one can trace your online activities, determine your physical location, or use your personal information without your consent.

Freedom Premium Services email and Web communications are encrypted and sent through the Freedom Network. Encryption scrambles your data, making it unintelligible to everyone except your intended recipient.

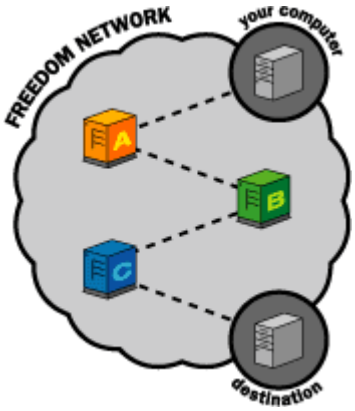
Stop Spam
Spam Blocker
Freedom Premium Services allow you to block unsolicited bulk email (spam) from ever reaching your inbox. Defend yourself against the annoyance of undesirable email clogging your inbox.

Encrypt all your communications
Private Encrypted Email Service
Essential for anyone who wants private and secure email, Zero-Knowledge's military-grade encryption works with your existing email program to ensure that no one, including your Internet Service Provider, can intercept and read your messages.

… And even more features to give you maximum protection
When subscribing to Freedom Premium Services, you also get access to additional features: a Personal Firewall to protect your PC from hackers; a Cookie Manager to prevent websites from tracking your surfing activities; a Form Filler / Password Manager to keep your personal information encrypted; an Ad Manager to block unwanted ads and speed up browsing; and a Keyword Alert to prevent any personal information from ever leaving your computer without your permission.

5. Mechanism Functionality – How does *Freedom* work?:

From:
http://www.freedom.net/products/premium/ps_technology.html?product=premium



The Freedom Network creates private routes between you and the destination computer.

Strong Encryption
When using Freedom Premium Services, all email and Web communications are encrypted and sent through the Freedom Network. Freedom's Premium Services use full-strength encryption (from 128-bits) on all incoming and outgoing Internet traffic. Encryption scrambles your data, making it illegible to everyone except your intended recipient. The entire encryption process is transparent, and operates in conjunction with your everyday Internet activities. This significantly reduces the risk of data theft or accidental leaks of sensitive information from your computer.

Freedom uses "Tunneling Encryption". In the diagram above, each of the three outlying encrypting nodes has its own public key. The traffic flow is encrypted three times: Source -> A ; A -> B ; B -> C A, B, and C are selected by pseudo-random means when the session is established. This method of encryption adds an additional degree of security: Node A knows only the Freedom client and Node B – Node B knows only Node A and Node C – Node C only knows Node B and the destination address.

TCP port 51107 is used during the setup phase to establish contact between the Freedom client and the Zero-Knowledge Network Information Query Server, which maintains a list of available encryption nodes. This contact occurs in cleartext and can be used as trigger material for an IDS.

The Dragon Detect of [ZKS: FREEDOM-SETUP]

The Freedom client setup with the Zero-Knowledge "encrypting cloud" is captured via Dragon's session reconstruction interface. Freedom Network Information Query Server communications to my network are in BLUE. My client responds in RED. After this exchange, all further session traffic is completely encrypted. Detection triggers are highlighted.

Sep 19, 2001

```
0 0.1.0 220 NODATA BINARY 0 "Zero-Knowledge Network Information
Query Server (NIQS) (Nov 25 2000, 01:39:28)"{D}{A}{D}{A}.{D}{A}{A}

0 7.1.0 mxquery nodata binary 0 description 0 BZIP2 keyqry{D}{A}{A}

0 7.1.0 220 NODATA BINARY 322 "Cached response"{D}{A}
{1}{C}{0}{0}{0}{0}{0}{0}{0}{0}{0}{1F} >
\{6}*{B5}{95}{C3}{AB}${89}{8E}{C9}{1}{9F}{B9}D{E0}{EB}{D3}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{95}{7}d+{E9}W{84}{D6}{7F}q{9A}{E}{8F}e-
{E2}{A6}P{1F}Z{1}{4}{0}{0}{0}{1}{6}{13}NIQS1{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}
{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{BA}
{1}{0}{5}{0}{2}{0}{1}{0}{A2}{2}{0}{0}{AA}{0}{0}{0}BZh11AY&SY}G3{B4}{0
}{0}{16}{7F}{D4}{F3}{98}@ @{2}{7F}{80}
l{9}({9A}{6}{8F}{9D}{A0}@{0}{92}{0}{0}{84}{80}{8}
{0}t"{A7}{B5}&{CA}m{10}4{C0}{86}{86}OiCC{C9}3{D4}{83}EODbd{C0}{13}
{4}`{2}0{8}{96}MpJO{D4}{EE}{BA}cc'{1A}a{0},{C}{10}{E4}{C8}t{8}{E1}{0}
{6}{C0}2({CC}{85}{8}{DA}{80}{9B}{CB}{DF}`7{A7}
{A6}{C8}{B2}b{{B1}0'r.{B8}A{85}N@{D8}{FC}{C}t|w{EC}\{E7}?3^{92}8'{0}@
`s{19}j%{E}{C8}{80}{F3}{15}q{1F}{E2}{EE}H{A7}{A}
{12}{F}{A8}{E6}v{80}{D}{A}.{D}{A}
```

6. Correlations:

No correlations were discovered. This is the first analysis of the Freedom product.

7. Evidence of Active Targeting:

The client must connect to the Zero-Knowledge network server by first using TCP port 51107. This constitutes Active Targeting in that use of Freedom is a conscious act.

8. Severity:

All estimates of severity in the use of Freedom are subjective because they are dependent on the network policies of the observer. For instance, a Freedom session originating from a .mil domain is subjectively of more concern that one originating from a .org site. Therefore, the full min-max range of zero to five must be assumed.

Target Criticality(TC):	0 (If network policy allows) 3 (Use from .com domain) 5 (Use from .mil domain)
Use Lethality(UL):	0 (Mechanism is not a vulnerability in itself, use allowed) 5 (Use in violation of policy, suspicious activity)
System Countermeasures(SC):	0 (Use allowed)
Network Countermeasures(NC):	0 (Use allowed, no blocking in place) 5 (Use not allowed, blocking in place)

Use Severity:

Case 1: Normal home use
(TC + UL) – (SC + NC) = (0 + 0) - (0 + 0) = 0

Case 2: .com use allowed
(TC + UL) – (SC + NC) = (3 + 0) - (0 + 0) = 3

Case 3: .com use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 0) = 8

Case 4: .com use violates policy, blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 5) = 3

Case 5: .mil use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (5 + 5) - (0 + 0) = 10

9. Defensive Recommendation:

In environments where the use of Freedom is not allowed, block TCP port 51107.

10. Test Question:

Name two anonymizing applications that could be used in series with Freedom.

(a) PGP and Safeweb.
(b) I'm the keeper of the cheese, and you're the lemon merchant, get it?
(c) The Anonymizer and PGP.
(d) Safeweb and The Anonymizer.

Answer: (d)

Correlation Postscriptum:

Zero-Knowledge Systems announce that the Freedom network will shut down on October 22, 2001.

http://slashdot.org/articles/01/10/04/1526256.shtml

***Anarchy Online* (*AO*)**

And now for some fun! It is certainly a thrill to a play a game online with thousands of others from all over the world in real-time, but in the .com world, it is almost always construed as resource misuse. For this reason, IDS signatures for these types of games are appropriate.

3. Probability the source address was spoofed:

In this case, the spoofing probability is zero because I generated this detect myself using the AO game client.
The game client requires the establishment of a TCP three-way handshake so the session source address cannot be spoofed.

4. Mechanism Description – What is *Anarchy Online*?:

Anarchy Online (AO) is the latest in a series of Massively Multiplayer Online Role-Playing Games, or MMORPG's. From http://www.anarchyonline.com/ :

> "Anarchy Online is set almost 30,000 years into the future, in a science-fiction world created uniquely for the game. Players will participate in the ongoing conflict between revolutionary rebels and the mighty Omni-Tek corporation, and ultimately play a key role in deciding the fate of the planet, Rubi-Ka."

5. Mechanism Functionality – How does *AO* work?:

The AO client and game server set up communications on TCP port 7500. After a secure login, lists of game servers are presented to the client for the player to choose from. When a server is selected, the game is invoked on the client system. The client-server setup phase is the only time cleartext is found the network traffic trace.

The Dragon Detect of [GAMES:ANARCHY-ONLINE-1]

AO client setup its game servers is captured via Dragon's session reconstruction interface. AO server communications to my network are in BLUE. There are no client responses in this trace. After this exchange, all further session traffic is indecipherable. Detection triggers are highlighted.

Sep 19, 2001

```
HTTP/1.1 200 OK{D}{A}

Date: Thu, 20 Sep 2001 00:22:54 GMT{D}{A}

Server: Apache/1.3.20 (Unix) mod_fastcgi/2.2.10 (mod_pcgi2/1.1.2;
PCGI/2.0a5) mod_perl/1.24_01 mod_ssl/2.8.4 OpenSSL/0.9.6{D}{A}

Cache-Control: max-age=60{D}{A}
Expires: Thu, 20 Sep 2001 00:23:54 GMT{D}{A}
Last-Modified: Thu, 23 Aug 2001 15:33:51 GMT{D}{A}
ETag: "13cf7-332-3b85225f"{D}{A}
Accept-Ranges: bytes{D}{A}
Content-Length: 818{D}{A}
Connection: close{D}{A}
Content-Type: text/plain{D}{A}{D}{A}

# This file holds all the timelines available at this time.{A}
# All lines starting with # is ignored by parser...{A}
#{A}{A}

STARTINFO{A}
description     =       Rubi-Ka 1{A}
```

```
connect         =       216.74.158.3{A}
ports           =       7500{A}
#url             =
http://www.anarchyonline.com/community/messages/index.html{A}

url             =       http://dimensions.anarchy-online.com/live-
dimensions/launcher/index.html{A}

version         =       12.5{A}
ENDINFO{A}{A}

STARTINFO{A}
description     =       Rubi-Ka 2{A}
connect         =       cm.d3.funcom.com{A}
#connect         =       216.74.158.232{A}
ports           =       7500{A}
#url             =
http://www.anarchyonline.com/community/messages/index.html{A}

url             =       http://dimensions.anarchy-online.com/live-
dimensions/launcher/index.html{A}

version         =       12.5{A}
ENDINFO{A}
```

6. Correlations:

No correlations were discovered. This is the first analysis of Anarchy Online.

7. Evidence of Active Targeting:

The AO client must connect to the game servers using TCP port 7500. This constitutes Active Targeting in that use of the AO client is a conscious act.

8. Severity:

All estimates of severity in the use of AO are subjective because they are dependent on the network policies of the observer. For instance, an AO session originating from a .mil domain is subjectively of more concern that one originating from a .org site. Therefore, the full min-max range of zero to five must be assumed.

Target Criticality(TC): 0 (If network policy allows)

3 (Use from .com domain)
5 (Use from .mil domain)

Use Lethality(UL): 0 (Mechanism is not a vulnerability in itself, use allowed)
5 (Use in violation of policy, suspicious activity)

System Countermeasures(SC): 0 (Use allowed)

Network Countermeasures(NC): 0 (Use allowed, no blocking in place)
5 (Use not allowed, blocking in place)

Use Severity:

Case 1: Normal home use
(TC + UL) – (SC + NC) = (0 + 0) - (0 + 0) = 0

Case 2: .com use allowed
(TC + UL) – (SC + NC) = (3 + 0) - (0 + 0) = 3

Case 3: .com use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 0) = 8

Case 4: .com use violates policy, blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 5) = 3

Case 5: .mil use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (5 + 5) - (0 + 0) = 10

9. Defensive Recommendation:

In environments where the online gaming is not allowed, block TCP port 7500 to prevent the AO game client from connecting with its game servers.

10. Test Question:

Can the use of AO be hidden inside another application, such as Freedom?

(a) Yes.
(b) Only if AO is tunneled through SSH.
(c) No.
(d) And he knows it!

Answer: (c)

SecureSHell

3. Probability the source address was spoofed:

In this case, the spoofing probability is zero because I generated this detect myself using the Secure Shell (SSH) Windows client.
The SSH client requires the establishment of a TCP three-way handshake so the session source address cannot be spoofed.

4. Mechanism Description – What does *SSH* do?:

From: http://www.ssh.com/products/ssh/features.cfm

Features of the SSH Secure Shell include:

- Protects all passwords and data
- Full replacement for telnet, rlogin, rsh, rcp, and ftp
- Fully integrated secure file transfer and file copying
- Really cool graphical user interface on Windows
- Automatic authentication of users, no passwords sent in cleartext to prevent the stealing of passwords
- Multiple strong authentication methods that prevent such security threats as spoofing identity
- Authentication of both ends of connection, the server and the client are authenticated to prevent identity spoofing, trojan horses, etc.
- Automatic authentication using agents to enable strong authentication to multiple systems with a single sign-on
- Transparent and automatic tunneling of X11 sessions

- Tunneling of arbitrary TCP/IP-based applications, such as e-mail
- Encryption and compression of data for security and speed
- Multiple built-in authentication methods, including passwords, public key, SecurID, and host-based authentication
- Support for PKI (Public Key infrastructure) and hardware tokens (e.g. smart cards)
- Multiple ciphers for encryption, including e.g. 3DES, Blowfish, Twofish and the AES candidate Rijndael

5. Mechanism Functionality – How does *SSH* work?:

The SSH client and server set up communications on TCP port 22 by default. The client-server setup phase is the only time cleartext is found the network traffic trace.

The Dragon Detect of [SSH:VERSION-1]

SSH client setup with its server is captured via Dragon's session reconstruction interface. SSH server communications to my network are in BLUE. My client responds in RED. After this exchange, all further session traffic is encrypted. Detection triggers are highlighted.

Sep 19, 2001

SSH-1.99-OpenSSH_2.9p1{A}{A}

SSH-1.99-2.4.0 SSH Secure Shell for Windows{A}{A}

{0}{0}{2}t{9}{14}{CB}{19};{B9}{E9}!{D0}eW{85}{FB}{97}{87}9{E9}{D8}{0}
{0}{0}=diffie-hellman-group-exchange-sha1,diffie-hellman-group1-

sha1{0}{0}{0}{7}ssh-dss{0}{0}{0}{96}aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc,rijndael128-cbc,rijndael192-cbc,rijndael256-cbc,rijndael-cbc@lysator.liu.se{0}{0}{0}{96}aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc,rijndael128-cbc,rijndael192cbc,rijndael256cbc,rijndaelcbc@lysator.liu.se{0}{0}{0}Uhmac-md5,hmac-sha1,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96{0}{0}{0}Uhmac-md5,hmac-sha1,hmac-ripemd160,hmac-ripemd160@openssh.com,hmacsha196,hmacmd596{0}{0}{0}{9}none,zlib{0}{0}{0}{9}none,zlib{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{0}{A}

{0}{0}{0}{E4}{6}{14}={FF}C{ED}{15}{FB}{E}{AC}{F3}x{BC}{B3}{2}{FC}{9C}{D7}{0}{0}{0}{1A}diffie-hellman-group1-sha1{0}{0}{0}{7}ssh-dss{0}{0}{0})3des-cbc,blowfish-cbc,twofish-cbc,arcfour{0}{0}{0})3des-cbc,blowfish-cbc,twofish-cbc,arcfour{0}{0}{0}{12}hmac-md5,hmac-sha1{0}{0}{0}{12}hmac-md5,hmac-sha1{0}{0}{0}{4}none{0}{0}{0}{4}none{0}{0}{0}{0}{0}{0}{0}{0}{1}{0}{0}{0}{0}\{E9}{F3}+,{0}

6. Correlations:

Numerous SSH correlations, including failed connection attempts and scanning for open SSH server ports are available at http://www.incidents.org/ A few example detects are provided below:

http://www.incidents.org/archives/intrusions/msg01659.html
http://www.incidents.org/archives/y2k/013001.htm

7. Evidence of Active Targeting:

The server destination port is configurable, but is most commonly found at port 22. This constitutes Active Targeting in that use of the SSH client is a conscious act.

8. Severity:

All estimates of severity in the use of SSH are subjective because they are dependent on the network policies of the observer. For instance, an SSH session originating from a .mil domain is subjectively of more concern that one originating from a .org site. Therefore, the full min-max range of zero to five must be assumed.

Target Criticality(TC): 0 (If network policy allows)
3 (Use from .com domain)
5 (Use from .mil domain)

Use Lethality(UL): 0 (Mechanism is not a vulnerability in itself, use allowed)
5 (Use in violation of policy, suspicious activity)

System Countermeasures(SC): 0 (Use allowed)

Network Countermeasures(NC): 0 (Use allowed, no blocking in place)
5 (Use not allowed, blocking in place)

Use Severity:

Case 1: Normal home use
(TC + UL) – (SC + NC) = (0 + 0) - (0 + 0) = 0

Case 2: .com use allowed
(TC + UL) – (SC + NC) = (3 + 0) - (0 + 0) = 3

Case 3: .com use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 0) = 8

Case 4: .com use violates policy, blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 5) = 3

Case 5: .mil use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (5 + 5) - (0 + 0) = 10

9. Defensive Recommendation:

TCP port 22 can be blocked to remove default SSH connectivity, but if the destination SSH server is run on a non-standard port, effective blocking will be problematic at best.

10. Test Question:

How can SSH client use be detected in an environment where SSH servers can run on any free TCP port?

(a) That's why he's gonna *kill* us! We gotta get outta here!
(b) SSH only runs on UDP ports.
(c) It can't.
(d) The signature must be applied to every TCP port.

Answer: (d)

***Pretty Good Privacy* (PGP)**

3. Probability the source address was spoofed:

In this case, the spoofing probability is zero because I generated this detect myself using by inserting a PGP-encrypted file into an email message using the Eudora client.
In reality the email source address could be spoofed using standard SMTP techniques.

4. Mechanism Description – What does *PGP* do?:

From http://www.pgp.com/products/mail-file-encryption/default.asp

"PGP encrypts, decrypts, signs and verifies files for either email or secure storage on your computer." (Note: There's a lot going on under PGP's hood, but that's another story for another time…)

Algorithms Supported:
Symmetric Key Encryption:
- 3DES
- CAST
- Twofish 256-bit

Public Key Encryption:
- RSA v4 up to 4096-bit
- DSS
- Diffie-Hellman

5. Mechanism Functionality – How does *PGP* work?:

The PGP client encrypts and signs a text file. That file is then pasted into an email client and sent to its recipient.

The Dragon Detect of [PGP-EMAIL]

The PGP email content is detected on any TCP port, although in practice ports 25 (SMTP) or 110 (POP3) are the most commonly used methods. The presence of PGP encrypted content is captured via Dragon's session reconstruction interface. POP3 server communications to my network are in BLUE. There is no client response in this trace. Detection triggers are highlighted.

Sep 23, 2001

```
X-Sender: MyUserName/smtp.ISP.net@pop3.norton.antivirus{D}{A}
X-Mailer: QUALCOMM Windows Eudora Version 5.1{D}{A}
Date: Sun, 23 Sep 2001 13:40:51 -0400{D}{A}
To: MyUserName@ISP.net{D}{A}
From: Randy Taylor  < MyUserName@ISP.net > {D}{A}
Subject: PGP test{D}{A}
Mime-Version: 1.0{D}{A}
Content-Type: text/plain; charset="us-ascii"; format=flowed{D}{A}
{D}{A}

-----BEGIN PGP MESSAGE-----{D}{A}
Version: PGP Personal Privacy 6.5.1{D}{A}{D}{A}
qANQR1DBwU4DJyxISp5ZwhAQCACNfWXEjiLckZeG/IGgRaNo85PpzWfqENb+oQ/+{D}{A
}
vgKFLXvtODkV2LtvPoeeGhHK66BUXWrPIhq7rnUW5pH3U+ggYixEl+VKrXrXn/9L{D}{A
}
q83JdQmF7sPwCzrsFtUkscdqC9Tsame8bVG2SI7FDblHPaxlPZs/fQ1Ktzvipls9{D}{A
}
TYtKyHXPBSMb9kNUCiVHbQLaTEf2xX8XG44GeUaV5J09OCR2JLaK0ha1KPv3datB{D}{A
}
kPPBHDsdVVNvtg5u683M3v4ji2oHh2M/9vccF9QKrzwZor221ArAUDkfk40EEoaa{D}{A
}
TSPYjDZf+eZWC39cUCseJiom1xx/Xw9vxZZANJFHdPSGWAkCB/9QfK0rljJSj9Y9{D}{A
}
bOFhwzpvZeYdnwuJ9TBLJZDLuRRX6Axp19lfz8HUNVQRT+mxpypt2TbFcaCR8jp1{D}{A
}
exBI6XgdIFTZpi8rElyZDFh1Eyx4Fm/iOA4bl+6r7MJywC1KcQm7u1h3K5wmL7Oi{D}{A
```

```
}
W7I/1W2UjplqxnoYrkWfz/mLj35QEV3ouVRUJYSG4w0vZCWsQjG8YNHg/SXjAP4n{D}{A
}
++ruV9ZNI5GTYdPbULZpDrvyEI9JSg/3ypA5TKNTAIFA3z4tY6eo1MSFAUpS/iqW{D}{A
}
tayCrAFjEA2W9voN+0LJ0a2XKyhKC90LDauO7TgvcM9dnS/sOZshR4RkkxIQozDW{D}{A
}
jpjUDdJwyYDSorNbhXtH0uo19EEy1tUYUvdnVr4BfKVh5IrsYhBrPVGa/9CjriaF{D}{A
}
/Tl3Bv5StBYZZTyeygjgMpHshzUX/z2dWGaxXzg8bQ8DeNcaPwYqN6lwNKyYRtj8{D}{A
}
70zZKM0RZryfxazzSx9LCXCnOluEdZDa+KEAontWQ2PT9p9m2nu8bw=={D}{A}
=stVU{D}{A}

-----END PGP MESSAGE-----{D}{A}

{D}{A}.{D}{A}
```

6. Correlations:

No correlations were discovered. This is the first analysis of PGP use.

7. Evidence of Active Targeting:

The presence of PGP encrypted content in an email message is a conscious act. While the source address can be spoofed, the destination address cannot – this supports a conclusion of Active Targeting.

8. Severity:

All estimates of severity in the use of PGP are subjective because they are dependent on the network policies of the observer. For instance, a PGP-encrypted message originating from a .mil domain is subjectively of more concern that one originating from a .org site. Therefore, the full min-max range of zero to five must be assumed.

Target Criticality(TC):	0 (If network policy allows) 3 (Use from .com domain) 5 (Use from .mil domain)
Use Lethality(UL):	0 (Mechanism is not a vulnerability in itself,

use allowed)
5 (Use in violation of policy, suspicious activity)

System Countermeasures(SC): 0 (Use allowed)

Network Countermeasures(NC): 0 (Use allowed, no blocking in place)
5 (Use not allowed, blocking in place)

Use Severity:

Case 1: Normal home use
(TC + UL) – (SC + NC) = (0 + 0) - (0 + 0) = 0

Case 2: .com use allowed
(TC + UL) – (SC + NC) = (3 + 0) - (0 + 0) = 3

Case 3: .com use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 0) = 8

Case 4: .com use violates policy, blocking in place
(TC + UL) – (SC + NC) = (3 + 5) - (0 + 5) = 3

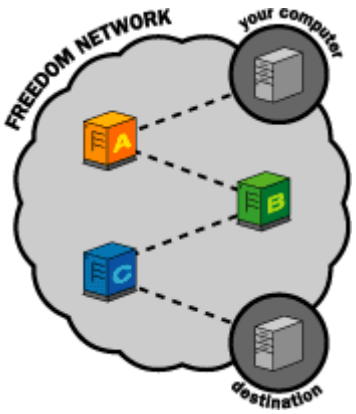
Case 5: .mil use violates policy, no blocking in place
(TC + UL) – (SC + NC) = (5 + 5) - (0 + 0) = 10

9. Defensive Recommendation:

There is no suitable method of completely blocking PGP message transmission.

10. Test Question:

Using the Freedom diagram below, where can a tunneled PGP message be detected?



(a) Between node C and the destination.
(b) Between nodes B and C.
(c) Yea! Before he lets loose the *marmosets* on us!
(d) Between the client and node A.

Answer: (a)

Assignment 3 – "Analyze This" Scenario

Intrusion Detection In Depth
GCIA Practical Assignment

Joseph Taylor
Version 2.9x

Executive Summary

- No direct evidence of customer system compromise was discovered.
- There was significant evidence suggesting that customer network security should be re-evaluated.
- Numerous services were found to open to probing from remote sources.
- Significant levels of contact were noted with external systems that have a history of unauthorized activity. There is no direct evidence, however, that customer system compromise occurred as a result.

Analysis Time Period Coverage

Analysis was broken out by individual days covering August 18 to August 22, 2001. The customer supplied no data for August 21, 2001

Data Sources

The customer supplied all analysis data from its Snort Intrusion Detection System. The data came in the form of Alert, OOS (Out-of-Spec), and Scan log files.

Citation of Sources

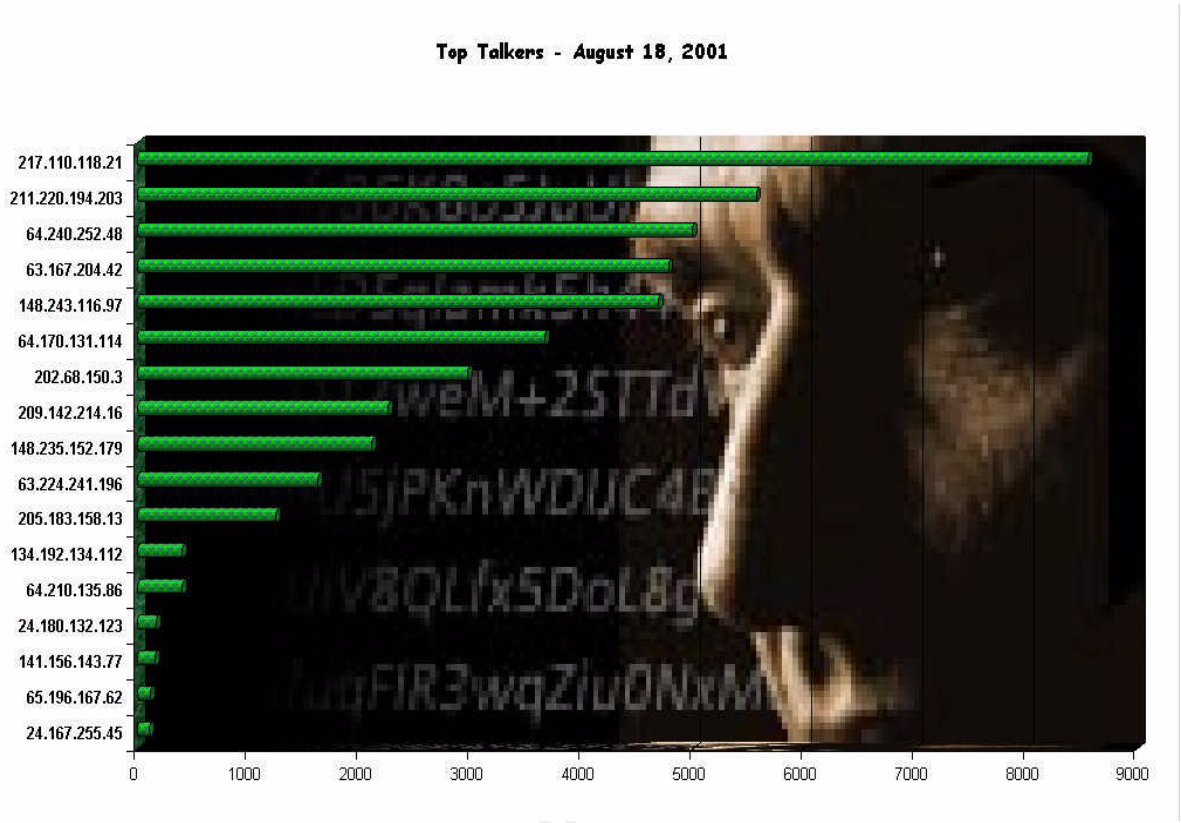
All reference sources of information are cited inline.

Analysis Method

1. "Top Talkers": Source IP addresses contacting or originating from the customer network were evaluated in terms of the total number of connections, per day.
2. All references to "MY.NET" were changed to "10.100" to enhance analysis capabilities.
3. "Top Five Top Talkers": The per-day five most active hosts were

evaluated for alert activity.

- Host Ownership: Network registration information was obtained for each host, when appropriate.
- A route trace to each host was conducted, when appropriate.
- Alert activity was correlated through various sources for relevant background vulnerability issues.
- Recommendations: Appropriate recommendations for mitigating the activity were discussed.



Analysis: The Top Five "Top Talkers" – August 18, 2001

217.110.118.21

The alerts from this system were comprised entirely of sweeps of the entire customer network (CIDR 10.100.0.0/16) for port 111 and some random ports in the 32771-32779 range. These scans are searching for open portmap/rpcbind services. In addition, during the course of these scans, twenty customer systems were probed for vulnerabilities in their RPC status services (*rpc.statd*). SecurityFocus describes *rpc.statd* as a server that implements the Network Status and Monitor RPC protocol. It is a component of the Network File System (NFS) architecture. *rpc.statd* attacks were noted in the 800-1009 and 32771-32779 port ranges. This indicates a mix of UNIX systems, including Linux, xBSD, and Solaris.

Ownership information for the 217.110.118 netblock is as follows:

```
inetnum:     217.110.118.0 - 217.110.118.255
netname:     DE-COLT-I3-INFORMATIONSTECHNOLOGIEN
descr:       I-3 INFORMATIONSTECHNOLOGIEN
descr:       BURGSTRASSE 49
descr:       49413 DINKLAGE
descr:       abuse? mailto:schlarmann@i-3.de
country:      DE
admin-c:     III3-RIPE
tech-c:      III3-RIPE
status:      ASSIGNED PA
notify:      ripemaster@de.colt.net
mnt-by:      DE-COLT-MNT
changed:     marcus.ruchti@colt.de 20010507
source:      RIPE
```

Colt is a telecommunications company headquartered in England. They provide services to most of the European Union countries, including Germany, the source of this traffic. Specifically, the source address appears to be in or near the Frankfurt area. This was determined by a route trace of the offending address and determining ownership of the last hop in the trace, ge1-2.ar06.fra1.DE.COLT-ISC.NET. This network is a subsidiary of Colt. An ownership query indicated:

```
Falk  Weinreich (template COCO-117741)
Falk.Weinreich@colt.de
Bleichstrasse 52
Frankfurt, -   63013   DE

Domain Name: colt-isc.net
Status:      production

Admin Contact:
  Falk Weinreich (COCO-117741) Falk.Weinreich@colt.de
  +49 69 95958 208 (FAX) +49 69 95958 100
Technical Contact:
      Hostmaster COLT Germany Hostmaster COLT Germany (COCO-
```

17363) hostmaster@de.colt.net
+49 69 95958 551 (FAX) +49 69 95958 6350

CORE Registrar: CORE-39

Record created: 2000-01-20 11:00:20 UTC by CORE-39
Record expires: 2002-01-20 04:10:57 UTC

Domain servers in listed order:

ns1.de.colt.net
ns0.de.colt.net

Database last updated on 2001-09-30 20:55:01 UTC

The most likely source of the traffic is a home system on either a dialup or broadband connection. This determination is made from the various pieces of ownership information – specifically that Colt and its subsidiaries are a telecommunications company that provides a variety of network-related services to EU countries. COLT-ISC.NET is likely to be a "last-mile" Internet Service Provider (ISP) for Germany in general and the Frankfurt area specifically.

Correlations:

There are two correlations noted in the SecurityFocus vulnerabilities database. Cross-correlations with MITRE's Common Vulnerabilities and Exposures (CVE) database are also provided where they exist:

1. http://www.securityfocus.com/bid/1480
   MITRE CVE: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0666

   Synopsis

   "A vulnerability exists in the rpc.statd program which is part of the nfs-utils packages, distributed with a number of popular Linux distributions. Because of a format string vulnerability when calling the syslog() function a malicious remote user can execute code as root."

2. http://www.securityfocus.com/bid/450

   MITRE CVE: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0493

   Synopsis

   "The vulnerability lies in (Sun Solaris) rpc.statd's ability to relay rpc calls to other rpc services without being validated by the access controls of the other rpc services. This can give the attacker the ability to redirect malicious rpc commands through rpc.statd (which runs as root) to services they may not normally have access to."

Scans for port 111 rank as the fifth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

Recommendation:

Block external access to TCP and UDP ports 111 and port range 32771-32779 at the customer network head-end routers.

211.220.194.203

The alerts from this system were comprised entirely of sweeps of the entire customer network (CIDR 10.100.0.0/16) for port 515. These scans are searching for open printer services. There is no indication that additional comprise methods were used after the scan.

Ownership information for the 211.220.194 netblock is as follows:

```
inetnum:      211.216.0.0 - 211.225.255.255
netname:      KORNET
descr:        KOREA TELECOM
descr:        KOREA TELECOM Internet Operating Center
country:      KR
admin-c:       DL276-AP
tech-c:       WK81-AP
remarks:      ****************************************
remarks:      Allocated to KRNIC Member.
remarks:      If you would like to find assignment
remarks:      information in detail please refer to
remarks:      the KRNIC Whois Database at:
remarks:      http://whois.nic.or.kr/english/index.html
remarks:      ****************************************
mnt-by:       MNT-KRNIC-AP
mnt-lower:    MNT-KRNIC-AP
changed:      hostmaster@apnic.net 20000901
changed:      hostmaster@apnic.net 20000912
changed:      hostmaster@apnic.net 20010627
source:       APNIC
```

KORNET is (South) Korea Telecom.

A traceroute to the offending system indicates the source of the traffic is a system within the Pusan area of South Korea (see highlighted area below):

Trace 211.220.194.203 ...

(Hops 1 through 10 redacted for brevity)

```
11      157.130.54.66   233ms  234ms  234ms  TTL:  0  (kt.co.kr-gw.customer.ALTER.NET ok)
12      211.216.216.2   247ms  234ms  234ms  TTL:  0  (glhub1-g5-0.kornet.net)
13      211.217.32.134          233ms   233ms   234ms   TTL:   0      (hh-c4-ge6.kornet.net)

14      168.126.109.14  234ms  234ms  234ms  TTL:  0  (pusan1-center4-2500M.kornet.net)
15      211.220.193.2   247ms  233ms  234ms  TTL:  0  (No rDNS)
16      211.220.194.203         234ms  247ms  233ms  TTL:235         (No rDNS)
```

The most likely source of the traffic is a home system on either a dialup or broadband connection, located in or near Pusan. This determination is made from ownership information – specifically that KORNET is a telecommunications company that provides a variety of network-related services throughout South Korea.

Correlations:

There are numerous correlations noted in the SecurityFocus vulnerabilities database. Only the two most recent printer vulnerabilities are listed. Cross-correlations with MITRE's Common Vulnerabilities and Exposures (CVE) database are also provided where they exist:

1. http://www.securityfocus.com/bid/3252
   MITRE CVE: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0670

   Synopsis

   "The BSD print protocol daemon, shipped with many systems, contains a remotely exploitable buffer overflow vulnerability. The daemon listens on tcp port 515 and facilitates printing over a network. It is often enabled by default."

2. http://www.securityfocus.com/bid/3274

Synopsis

"The print protocol daemon, 'in.lpd' (or 'lpd'), shipped with Solaris may allow for remote attackers to execute arbitrary commands on target hosts with superuser privileges."

Scans for port 515 rank as the eighth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

Recommendation:

Block external access to TCP and UDP port 515 at the customer network head-end routers.

64.240.252.48

The alerts from this system were comprised entirely of sweeps of the entire customer network (CIDR 10.100.0.0/16) for port 111. These scans are searching for open portmap/rpcbind services.

Ownership information for the 64.240.252 netblock is as follows:

Lloyd Lamont Design, Inc / Net2000 (NETBLK-SAVV-LLOYD-L2)
500 Grove Street, 3rd Floor
Herndon, VA 22170
US

Netname: SAVV-LLOYD-L2
Netblock: 64.240.252.0 - 64.240.252.255

Coordinator:

Somers, Nancy (NS107-ARIN) nsomers@net2000.com
(703)654-2943

Record last updated on 19-Apr-2000.
Database last updated on 29-Sep-2001 23:14:31 EDT.

From the company's home page at http://www.lld.com:

> "LLD, a certified small disadvantaged business (SDB), is a multi-faceted organization committed to the development and application of new technologies and innovations in the public and private sectors."

Given the company's description, it is probable that their network security is less than optimal. It is reasonable to assume remote attackers have compromised the lld.com network.

Correlations:

Scans for port 111 rank as the fifth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

This scan did not contact any other port except 111. The source ports used in the scanning effort were relatively sequential and increasing. The most common tool used for this type of scan is nmap.

Recommendation:

Block external access to TCP and UDP ports 111 and port range 32771-32779 at the customer network head-end routers.

63.167.204.42

The alerts from this system were comprised entirely of sweeps of the entire customer network (CIDR 10.100.0.0/16) for port 111. These scans are searching for open portmap/rpcbind services.

Ownership information for the 63.167.204 netblock is as follows:

> KORKSOFT (NETBLK-FON-106796134479193)
> 6630 SPRING GARDEN RUN
> LAKE WORTH, FL 33463
> US
>
> Netname: FON-106796134479193
> Netblock: 63.167.204.0 - 63.167.207.255
>
> Coordinator:
> KORKIN, JASON (JK1005-ARIN)
> HOSTMASTER@KORSOFT.COM
> (603)672-1246
>
> Record last updated on 05-Jun-2001.
> Database last updated on 29-Sep-2001 23:14:31 EDT.

Korksoft is a small web site hosting and design firm. By attempting to connect to 63.167.204.42's web server, I found that it is owned by "booksellersolutions.com".

From the company's information page at http://www.booksellersolutions.com/cgi-bin/index/products.html :

> "booksellersolutions.com offers an array of complete integrated services to help you quickly and affordably set up your own online bookstore under your own domain (www.yourbookstore.com)."

Ownership information for booksellersolutions.com is as follows:

> booksellersolutions.com (BOOKSELLERSOLUTIONS-DOM)

2141 Mission Street Suite 301
San Francisco, CA 94110
US

Domain Name: BOOKSELLERSOLUTIONS.COM

Administrative Contact, Billing Contact:
Lozier, Luke (LL7290) luke@BOOKSELLERSOLUTIONS.COM
booksellersolutions.com
2141 Mission Street Suite 301
San Francisco, CA 94110
+1 415 554-0568 (FAX) +1 415 252-8464
Technical Contact:
Korkin, Jason (JK7871) jkorkin@KORKSOFT.COM
KORKSOFT
KORKSOFT 8 Olde Bedford Way
Bedford, NH 03110
(603) 472-8262 (FAX) (603) 472-8262

Record last updated on 06-Sep-2001.
Record expires on 29-Sep-2002.
Record created on 29-Sep-1999.
Database last updated on 30-Sep-2001 05:01:00 EDT.

Domain servers in listed order:

NS1.BOOKSELLERSOLUTIONS.COM 63.167.204.24
NS2.BOOKSELLERSOLUTIONS.COM 63.167.205.16

A route trace indicates that the offending host is located in the Southeastern US. See the highlighted area below:

(Hops 1 through 9 redacted for brevity)

10 144.232.9.198	55ms	41ms	55ms	TTL: 0	(**sl-bb20-atl-10-1.sprintlink.net**)
11 144.232.12.238	55ms	138ms	233ms	TTL: 0	(No rDNS)
12 144.223.47.74	97ms	110ms	82ms	TTL: 0	(sl-korksoft-2-0.sprintlink.net)
13 63.167.204.42	96ms	96ms	83ms	TTL:243	(No rDNS)

The route to the booksellersolutions.com web site goes through Atlanta (*atl* in the highlighted host above). The most likely location for the web site is at

the Korksoft "web farm" in Lake Worth, Florida. It is reasonable to assume that the booksellersolutions.com web site was compromised remotely and is being used to conduct scans against the customer network.

Correlations:

Scans for port 111 rank as the fifth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

This scan did not contact any other port except 111. The source ports used in the scanning effort were relatively sequential and increasing. The most common tool used for this type of scan is *nmap*.

Recommendation:

Block external access to TCP and UDP ports 111 and port range 32771-32779 at the customer network head-end routers.

148.243.116.97

The alerts from this system were comprised entirely of sweeps of the entire customer network (CIDR 10.100.0.0/16) for port 111. These scans are searching for open portmap/rpcbind services.

Ownership information for the 148.243.116 netblock is as follows:

Avantel, S.A. (NETBLK-AVANTEL-BL11)
Vasconcelos 130 ote
San Pedro, Nuevo Leon 66267
MX

Netname: AVANTEL-BL11
Netblock: 148.243.0.0 - 148.243.255.255
Maintainer: AVAN

Coordinator:
Administrator, Noc (NA83-ARIN) noc@AVANTEL.NET.MX
(8) 156 3065

Domain System inverse mapping provided by:

DNS1.AVANTEL.NET.MX 200.33.213.66
DNS2.AVANTEL.NET.MX 200.33.209.66

Record last updated on 01-May-2001.
Database last updated on 29-Sep-2001 23:14:31 EDT.

The IP address resolves via DNS to *monalisa.nsmex.com*
Ownership of the nsmex.com domain is as follows:

NET SOLUTIONS (NSMEX-DOM)
Presas Salinillas 370. 3er Piso
MEXICO, CITY 11550
MX

Domain Name: NSMEX.COM

Administrative Contact:
HARARI, RAFAEL (RXH96) rafa@netmex.com
HARARI,RAFAEL
Alfredo Musset 38
MEXICO, MEXICO DF 11550
MX
5503 4003 123 123 1234
Technical Contact:
BISSU, MOISES (MBT458) bissu@NETSOLUTIONS.COM.MX
MOISES BISSU
Blvd. M Avila Camacho #681 L B-10
MEXICO CITY
D.F.
11220
MX
525 1489888 (FAX) 123 123 1234
Billing Contact:
NET SOLUTIONS-WN-CAAG (NS2186-ORG)
no.valid.email@worldnic.net
NET SOLUTIONS-WN-CAAG
Presas Salinillas 370. 3er Piso
MEXICO
MX
525 5034003 fax: 525 5034003

Record last updated on 09-Oct-2000.
Record expires on 27-Apr-2002.
Record created on 27-Apr-2000.
Database last updated on 30-Sep-2001 05:01:00 EDT.

Domain servers in listed order:

DNS1.NSMEX.COM 148.243.116.9
DNS2.NSMEX.COM 148.243.116.8

The *nsmex.com* domain is owned by Net Solutions Mexico, located in Mexico City. From their information page at

http://www.nsmex.com/menu/english/english.shtml:

> "**Net Solutions** offers a great variety of services focusing on high-quality and service. Dial-up connection service, leased lines, DirecPC, Lan-modems, Server Hosting, Web Hosting, Web Design, E-commerce, E-mail services, Domain Registration, Network Creation and Administration, and much more."

The offending host, *monalisa*, is not at the same address as the Net Solutions Mexico web site (148.243.116.97 vs. 148.243.116.8), but they are on the same subnetwork. There is not enough information to determine whether or not the compromise of *monalisa* originated local to the Mexico City area or was accomplished by remote means.

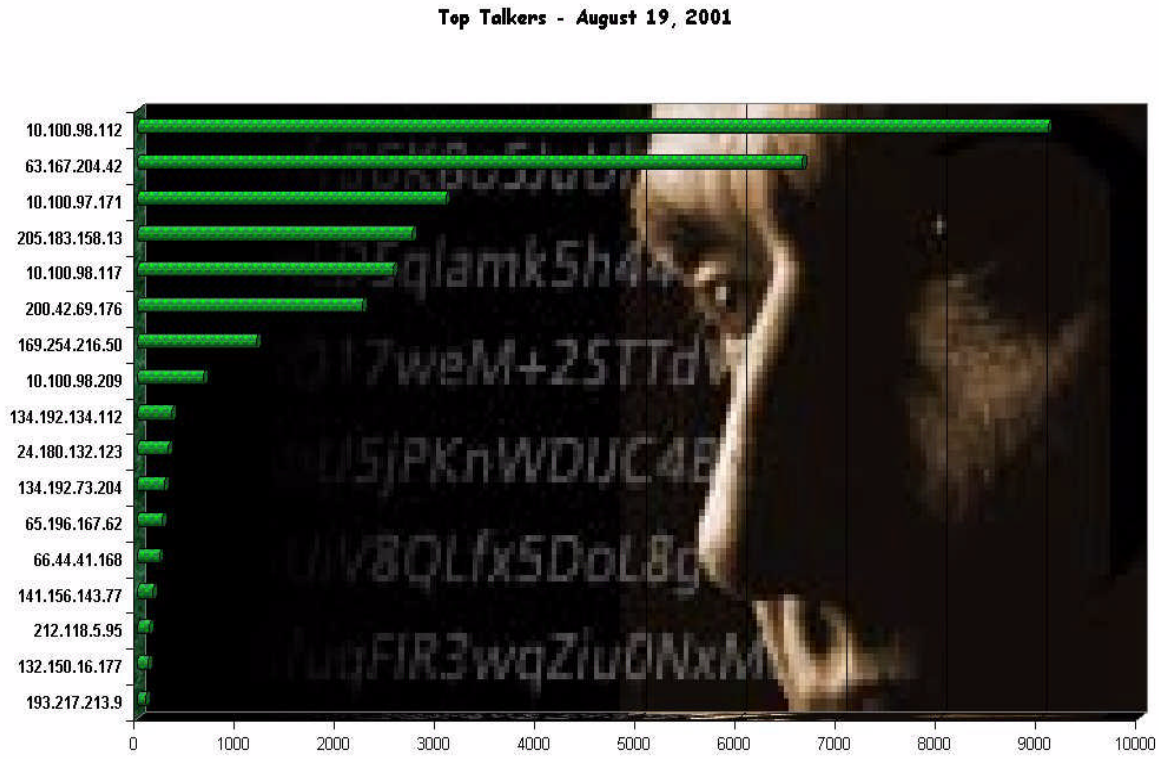
Correlations:

Scans for port 111 rank as the fifth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

This scan did not contact any other port except 111. The source ports used in the scanning effort were relatively sequential and increasing. The most common tool used for this type of scan is *nmap*.

Recommendation:

Block external access to TCP and UDP ports 111 and port range 32771-32779 at the customer network head-end routers.



Analysis: The Top Five "Top Talkers" – August 19, 2001

10.100.98.112

The alerts from this system were comprised entirely of activity correlating to a port scan for the SubSeven Trojan horse, which normally runs on TCP port 27374. The scan originates in the customer's network. The scan sweeps numerous external networks not owned by the customer. A sample of five are listed below:

12.25.197-198.0/24	Bowman Capital Management San Mateo, CA
12.98.197.0/24	AT&T (DSL Network) Middletown, NJ
66.25.200.0/24	Roadrunner (DSL Network)

	Herndon, VA
142.163.201-202.0/24	Stentor National Integrated Communications Network Ottawa, Canada
146.172.200-201.0/24	TBK A.S. Oslo, Norway

It is virtually certain these sweeps are unauthorized activity, given their breadth. Immediate action should be taken to terminate the scan and determine who is responsible for it. There is not enough data in the multi-day sample provided to ascertain if this host has been compromised by remote means, or if a customer employee or contractor is conducting the activity.

Correlations:

1. http://www.dshield.org/ports/port27374.html

   Synopsis

   "27374 is one of the default ports of the BackDoor-G2.svr.gen trojan, more commonly known as SubSeven. It is the current (as of May 2001) trojan of choice for most DDoS attacks and clone attacks on specific services, such as IRC."

2. http://www.symantec.com/avcenter/venc/data/backdoor.subseven.html

   Synopsis

   "Backdoor.SubSeven is a Trojan horse, similar to Netbus or Back Orifice. It enables unauthorized people to access your computer over the Internet without your knowledge."

Scans for port 27374 rank as the ninth most common at Incidents.Org

(http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

Recommendation:

Block external access to all unnecessary TCP and UDP ports at the customer network head-end routers.

63.167.204.42

The alerts from this system were comprised entirely of sweeps of the entire customer network (CIDR 10.100.0.0/16) for port 111 and some random ports in the 32771-32779 range. These scans are searching for open portmap/rpcbind services. In addition, during the course of these scans, twenty customer systems were probed for vulnerabilities in their RPC status services (*rpc.statd*). SecurityFocus describes *rpc.statd* as a server that implements the Network Status and Monitor RPC protocol. It is a component of the Network File System (NFS) architecture. *rpc.statd* attacks were noted in the 800-1009 and 32771-32779 port ranges. This indicates a mix of UNIX systems, including Linux, xBSD, and Solaris.

63.167.204.42 appeared in the August 18, 2001 "Top Talkers" list, performing the same types of scans. The difference between the two listings is that *rpc.statd* probes were not identified in the August 18, 2001 scans. Please see the August 18, 2001 entry for netblock ownership details.

Correlations:

There are two correlations noted in the SecurityFocus vulnerabilities database. Cross-correlations with MITRE's Common Vulnerabilities and Exposures (CVE) database are also provided where they exist:

1. http://www.securityfocus.com/bid/1480
   MITRE CVE: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0666

Synopsis

"A vulnerability exists in the rpc.statd program which is part of the nfs-utils packages, distributed with a number of popular Linux distributions. Because of a format string vulnerability when calling the syslog() function a malicious remote user can execute code as root."

2. http://www.securityfocus.com/bid/450

MITRE CVE: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0493

Synopsis

"The vulnerability lies in (Sun Solaris) rpc.statd's ability to relay rpc calls to other rpc services without being validated by the access controls of the other rpc services. This can give the attacker the ability to redirect malicious rpc commands through rpc.statd (which runs as root) to services they may not normally have access to."

Scans for port 111 rank as the fifth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

Recommendation:

Block external access to TCP and UDP ports 111 and port range 32771-32779 at the customer network head-end routers.

10.100.97.171

The alerts from this system were comprised entirely of activity correlating to a port scan for the SubSeven Trojan horse, which normally runs on TCP port 27374. The scan originates in the customer's network. The scan sweeps numerous external networks not owned by the customer. A sample of three are listed below:

24.65.141-142.0/24	Shaw Fiberlink, LTD Calgary, Alberta, Canada
66.25.200.0/24	Roadrunner (DSL Network) Herndon, VA
142.163.201-202.0/24	Stentor National Integrated Communications Network Ottawa, Canada

It is virtually certain these sweeps are unauthorized activity, given their breadth. Immediate action should be taken to terminate the scan and determine who is responsible for it. There is not enough data in the multi-day sample provided to ascertain if this host has been compromised by remote means, or if a customer employee or contractor is conducting the activity.

Correlations:

1. http://www.dshield.org/ports/port27374.html

   Synopsis

   "27374 is one of the default ports of the BackDoor-G2.svr.gen trojan, more commonly known as SubSeven. It is the current (as of May 2001) trojan of choice for most DDoS attacks and clone attacks on specific services, such as IRC."

2. http://www.symantec.com/avcenter/venc/data/backdoor.subseven.html

   Synopsis

"Backdoor.SubSeven is a Trojan horse, similar to Netbus or Back Orifice. It enables unauthorized people to access your computer over the Internet without your knowledge."

Scans for port 27374 rank as the ninth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

Recommendation:

Block external access to all unnecessary TCP and UDP services at the customer network head-end routers.

205.183.158.13

The alerts from this system were comprised entirely of activity correlating to port scans for Simple Network Management Protocol (SNMP) services, which normally runs on TCP and UDP port 161. The majority of the customer network (CIDR 10.100.0.0/16) was scanned. There is no evidence that compromise attempts against SNMP services were attempted.

Ownership information for the 205.183.158 netblock is as follows:

Cash America (NETBLK-CASHAM2-158-29)
6100 Western Place
Fort Worth, TX 76107
US

Netname: CASHAM2-158-29
Netblock: 205.183.158.0 - 205.183.158.255

Coordinator:
Smiley, Melody (MS55-ARIN) msmiley@CASHAM.COM

817-390-9293

Record last updated on 30-Jun-2000.
Database last updated on 29-Sep-2001 23:14:31 EDT.

Correlations:

Scans for port 161 rank as the tenth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

Recommendation:

Block external access to TCP and UDP port 161 at the customer network head-end routers.

10.100.98.117

The alerts from this system were comprised entirely of activity correlating to a port scan for the SubSeven Trojan horse, which normally runs on TCP port 27374. The scan originates in the customer's network. The scan sweeps numerous external networks not owned by the customer. A sample of two are listed below:

24.156.50-51.0/24	Rogers@Home(Cable Modem Network) Toronto, Ontario, Canada
24.16.50-51.0/24	@Home Network (Cable Modem Net) Redwood City, CA

It is virtually certain these sweeps are unauthorized activity, given their breadth. Immediate action should be taken to terminate the scan and

determine who is responsible for it. There is not enough data in the multi-day sample provided to ascertain if this host has been compromised by remote means, or if a customer employee or contractor is conducting the activity.

Correlations:

1. http://www.dshield.org/ports/port27374.html

   Synopsis

   "27374 is one of the default ports of the BackDoor-G2.svr.gen trojan, more commonly known as SubSeven. It is the current (as of May 2001) trojan of choice for most DDoS attacks and clone attacks on specific services, such as IRC."

2. http://www.symantec.com/avcenter/venc/data/backdoor.subseven.html

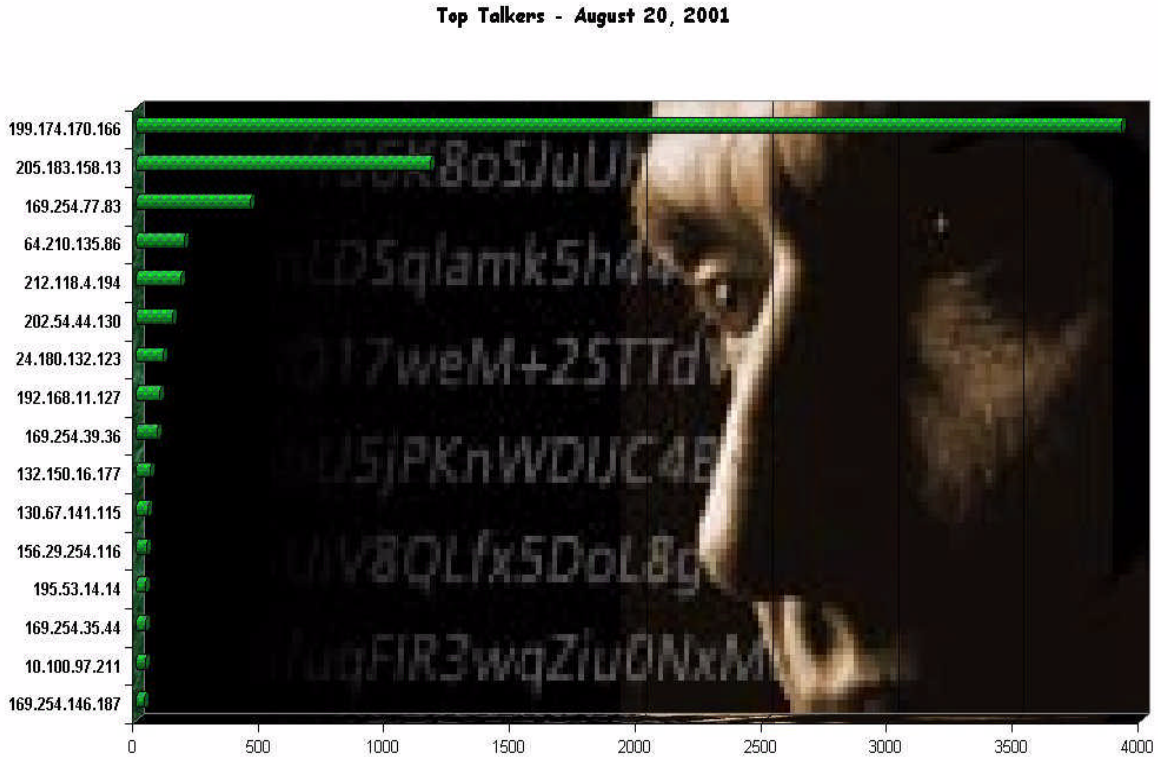
   Synopsis

   "Backdoor.SubSeven is a Trojan horse, similar to Netbus or Back Orifice. It enables unauthorized people to access your computer over the Internet without your knowledge."

Scans for port 27374 rank as the ninth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

Recommendation:

Block external access to all unnecessary TCP and UDP services at the customer network head-end routers.



Analysis: The Top Five "Top Talkers" – August 20, 2001

199.174.170.166

The alerts from this system were comprised entirely of sweeps of the entire customer network (CIDR 10.100.0.0/16) for port 515. These scans are searching for open printer services. There is no indication that additional comprise methods were used after the scan.

Ownership information for the 211.220.194 netblock is as follows:

EarthLink, Inc. (NET-EARTHLINK2000-C)
3100 New York Drive
Pasadena, CA 91107

US

Netname: EARTHLINK2000-C
Netblock: 199.174.0.0 - 199.174.255.255
Maintainer: ERMS

Coordinator:
Earthlink Network, Domain Administrator (DAE4-ARIN)
arinpoc@corp.earthlink.net
626-296-2400 (FAX) 626-296-5113

Domain System inverse mapping provided by:

ITCHY.MINDSPRING.NET 207.69.200.210
SCRATCHY.MINDSPRING.NET 207.69.200.211

Record last updated on 20-Apr-2000.
Database last updated on 29-Sep-2001 23:14:31 EDT.

A route trace indicates that the offending host is located in the Chicago area. See the highlighted areas below:

(Hops 1 through 11 redacted for brevity)

12 205.215.1.78	*	50ms	50ms TTL: 0	(Earthlink-45.pr1.**Chicago1**.**IL**.us.netrail.net)
13 207.69.219.163	60ms	*	50ms TTL: 0	(arc-9a.**chi**.mindspring.net ok)
14 199.174.170.166	171ms	150ms	151ms TTL:109	(user-33qtal6.**dial up**.mindspring.com)

The most likely source of the traffic is a home dialup connection, located in or near Chicago. This determination is made from ownership information – specifically that Mindspring is a nationally-known ISP that provides a variety of network-related services throughout the US.

Correlations:

There are numerous correlations noted in the SecurityFocus vulnerability database. Only the two most recent printer vulnerabilities are listed. Cross-correlations with MITRE's Common Vulnerabilities and Exposures (CVE) database are also provided where they exist:

1. http://www.securityfocus.com/bid/3252
   MITRE CVE: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0670

   Synopsis

   "The BSD print protocol daemon, shipped with many systems, contains a remotely exploitable buffer overflow vulnerability. The daemon listens on tcp port 515 and facilitates printing over a network. It is often enabled by default."

2. http://www.securityfocus.com/bid/3274

   Synopsis

   "The print protocol daemon, 'in.lpd' (or 'lpd'), shipped with Solaris may allow for remote attackers to execute arbitrary commands on target hosts with superuser privileges."

Scans for port 515 rank as the eighth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

Recommendation:

Block external access to TCP and UDP port 515 at the customer network head-end routers.

205.183.158.13

The alerts from this system were comprised entirely of activity correlating to port scans for Simple Network Management Protocol (SNMP) services, which normally runs on TCP and UDP port 161. The majority of the customer network (CIDR 10.100.0.0/16) was scanned. There is no evidence that compromise attempts against SNMP services were attempted.

205.183.158.13 appeared in the August 19, 2001 "Top Talkers" list, performing the same types of scans.

Correlations:

Scans for port 161 rank as the tenth most common at Incidents.Org (http://www.incidents.org/), a co-operative intrusion detection database operated by SANS.

Recommendation:

Block external access to TCP and UDP port 161 at the customer network head-end routers.

169.254.77.83 , 64.210.135.86

The traffic from this host did not affect the customer site at all. It was listed as port 137 UDP traffic sourcing from and destined to hosts outside the customer network. There were no corresponding contacts to TPC or UDP port 139, which would represent attempts to attach to customer network Windows file shares.

Correlations:

1. http://www.dshield.org/ports/port137.html

   Synopsis

   "Windows uses it's own system to translate IP addresses into Windows names. Many of these probes may just be caused by this quirk of Windows. A probe of port 137 should not be seen as evidence of an attack. However, if you see (sic) simultanous access to port 139, you should be alarmed. In this case, someone may actually try to connect to your PC and access its shared resources."

Recommendation:

Filter TCP and UDP port 137 at the customer network gateways router(s) to eliminate alerts when both source and destination IP addresses do not correspond to customer networks.

212.118.4.194

This host contacted several customer network machines on UDP port 137. There were no corresponding contacts to TPC or UDP port 139, which would represent attempts to attach to customer network Windows file shares.

Ownership information for the 212.118.4 netblock is as follows:

```
inetnum:     212.118.0.0 - 212.118.10.255
netname:     NETS-NETWORK
descr:       National Equipment & Technical Services,
descr:       Internet Service Provider
descr:       Located in Amman-Jordan
country:     JO
admin-c:     MSJ7-RIPE
tech-c:      RS37-RIPE
status:      ASSIGNED PA
notify:      admin@nets.com.jo
mnt-by:      RIPE-NCC-NONE-MNT
changed:     hostmaster@ripe.net 19981026
changed:     rami@nets.com.jo 19990501
changed:     rami@nets.com.jo 20000416
changed:     rami@nets.com.jo 20000920
changed:     rami@nets.com.jo 20010206
changed:     rami@nets.com.jo 20010617
source:      RIPE
```

This contact from Jordan is at most, suspicious, but the supplied data does not indicate additional compromise activity. Web sites using Microsoft server software routinely and frequently query port 137 during their use, but there is nothing deliberately hostile in this.

Correlations:

1. http://www.dshield.org/ports/port137.html

   Synopsis

   "Windows uses it's own system to translate IP addresses into Windows names. Many of these probes may just be caused by this quirk of Windows. A probe of port 137 should not be seen as evidence of an attack. However, if you see (sic) simultanous access to port 139, you should be alarmed. In this case, someone may actually try to connect to your PC and access its shared resources."

Recommendation:

None. Not enough evidence to determine intent.

Analysis: The Top Five "Top Talkers" – August 22.2001

10.100.217.18, 10.100.151.63, 10.100.163.100

All three of these customer systems were contacted by external systems on SANS "Watchlist". The Watchlist contacts are as follows:

Customer System	Watchlist System	Port
10.100.217.18	212.179.27.6	6346
	212.179.34.114	
	212.179.58.194	
10.100.151.63	212.179.27.6	1214
10.100.163.100	159.226.41.166	1604

Ownership information for the 212.179.27 netblock is as follows:

route: 212.179.0.0/17
descr: ISDN Net Ltd.
origin: AS8551
notify: hostmaster@isdn.net.il
mnt-by: AS8551-MNT
changed: hostmaster@isdn.net.il 19990610
source: RIPE

inetnum: 212.179.27.0 - 212.179.27.3
netname: GALIL-ENGINEERING
descr: GALIL-ENGINEERING-SERIAL
country: IL
admin-c: NP469-RIPE
tech-c: NP469-RIPE
status: ASSIGNED PA
notify: hostmaster@isdn.net.il
mnt-by: RIPE-NCC-NONE-MNT
changed: hostmaster@isdn.net.il 20000106
source: RIPE

Ownership information for the 212.179.34 netblock is as follows:

inetnum: 212.179.34.0 - 212.179.34.31
netname: TOTEM-SYSTEMS
descr: TOTEM-SYSTEMS-LAN
country: IL
admin-c: NP469-RIPE
tech-c: NP469-RIPE
status: ASSIGNED PA
notify: hostmaster@isdn.net.il
mnt-by: RIPE-NCC-NONE-MNT
changed: hostmaster@isdn.net.il 20000106
source: RIPE

Ownership information for the 212.179.58 netblock is as follows:

inetnum: 212.179.58.0 - 212.179.58.255
netname: NV-PICTUREVISION
descr: network
country: IL
admin-c: NP469-RIPE
tech-c: NP469-RIPE
status: ASSIGNED PA
notify: hostmaster@isdn.net.il
mnt-by: RIPE-NCC-NONE-MNT
changed: hostmaster@isdn.net.il 20000229
source: RIPE

Ownership information for the 159.226.41 netblock is as follows:

The Computer Network Center Chinese Academy of Sciences (NET-NCFC)
P.O. Box 2704-10,
Institute of Computing Technology Chinese Academy of Sciences
Beijing 100080, China
CN

Netname: NCFC
Netblock: 159.226.0.0 - 159.226.255.255

Coordinator:
Qian, Haulin (QH3-ARIN) hlqian@NS.CNC.AC.CN
+86 1 2569960

Domain System inverse mapping provided by:

NS.CNC.AC.CN 159.226.1.1
GINGKO.ICT.AC.CN 159.226.40.1

Record last updated on 25-Jul-1994.
Database last updated on 29-Sep-2001 23:14:31 EDT.

Port 6346 is most commonly used for *Gnutella*, a peer-to-peer (P2P) file sharing application associated with music files in the MP3 format. Port 1214 is associated with *KaZaA* or *Morpheus*, which are also P2P applications.

Port 1604 is used for Citrix ICA (a.k.a. *icabrowser*), which uses Windows Terminal Services fetaures.

The P2P accesses may be benign, but *icabrowser* access attempts may not be.

Correlations:

1. Port 1604 (*icabrowser*)

   http://www.netice.com/Advice/Exploits/Ports/1604/default.htm

   Synopsis

   "Citrix ICA is a remote Windows terminal program. The software was licensed by Microsoft and included in the Microsoft "Windows Terminal Server" product for Windows NT 4.0. The technology was later included as an integral part of Windows 2000 Server."

2. Port 6346 (*Gnutella*)

   http://www.securityfocus.com/bid/3267

   Synopsis

   "Gnut is a free, open-source console-based Gnutella file-sharing client for Microsoft Windows and Linux systems. A problem exists with Gnut's web interface. Webfrontend allows users to perform searches, but when the results of a search are returned the interface will not strip HTML tags from filenames. An attacker could exploit this issue by embedding script code in a filename. Webfrontend is often viewed on "localhost", so therefore the malicious script code may also be executed in the system context rather than Internet context, circumventing the browser-based zone security settings."

3. Port 1214 (KaZaA, Morpheus)

   http://www.securityfocus.com/bid/3125

   Synopsis

   "It is possible to specify a folder to share out with other Morpheus or KaZaA users, it is also possible to specify a folder where downloaded files will be saved to. A flaw exists in the Media sharing component that could enable a user to view the contents of the folder specifed for download files, if file sharing is enabled. "

Recommendation:

Block all access to the customer network from networks on the SANS Watchlist. Review P2P use against customer network AUP's. Audit customer systems for vulnerabilities related to Citrix and Microsoft Terminal Services.

164.107.3.40, 233.25.109.2

The traffic from these hosts did not affect the customer site at all. It was listed as port 137 UDP traffic sourcing from and destined to hosts outside the customer network. There were no corresponding contacts to TPC or UDP port 139, which would represent attempts to attach to customer network Windows file shares.

Correlations:

1. http://www.dshield.org/ports/port137.html

   Synopsis

   "Windows uses it's own system to translate IP addresses into Windows names. Many of these probes may just be caused by this quirk of Windows. A probe of port 137 should not be seen as evidence of an attack. However, if you see (sic) simultanous access to port 139, you should be alarmed. In this case, someone may actually try to connect to your PC and access its shared resources."

Recommendation:

Filter TCP and UDP port 137 at the customer network gateways router(s) to eliminate alerts when both source and destination IP addresses do not correspond to customer networks.