



Interested in learning more about security?

SANS Institute InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Case Study - Windows 2000 ISA Proxy Server Authentication Inside a DMZ

This paper describes the investigation process and implementation of IPSec policies to manage a wide range of communication traffic between two Windows 2000 servers. The use of IPSec allowed us to configure an ISA Proxy server to authenticate user login information back to an internal DC through a high security firewall. Extra configuration involving name resolution and security lockdowns completed our solution for a secure and functional proxy implementation. One of the most difficult aspects of firewall design is bal...

Copyright SANS Institute
Author Retains Full Rights

AD

Build your business' breach action plan.

START NOW

 **LifeLock**
BUSINESS SOLUTIONS

No one can prevent all identity theft. © 2016 LifeLock, Inc. All rights reserved. LifeLock and the LockMan logo are registered trademarks of LifeLock, Inc.

Case Study - Windows 2000 ISA Proxy Server Authentication Inside a DMZ.

GSEC Practical Assignment Version 1.4

Option 2 – Case Study

Michael Kerr

6/12/2002

Abstract

This paper describes the investigation process and implementation of IPSec policies to manage a wide range of communication traffic between two Windows 2000 servers. The use of IPSec allowed us to configure an ISA Proxy server to authenticate user login information back to an internal DC through a high security firewall. Extra configuration involving name resolution and security lockdowns completed our solution for a secure and functional proxy implementation.

One of the most difficult aspects of firewall design is balancing security with functionality. In the absence of business requirements it is a straightforward task to design a highly secure firewall and DMZ environment, however, providing functionality to the organisation being protected by the perimeter network is every bit as important as providing high security. This case study is not going to discuss the broader concepts of DMZ design, instead I want to focus on a particular business requirement of a newly implemented perimeter network, how we studied the problem and the actions we took to provide the desired functionality. The topic I've chosen is interesting and hopefully will be valuable to others who would like to get extra value from their internal Active Directory service.

The Challenge

Our chosen DMZ topology is a traditional high security layout; it's shown in figure 1. Utilizing two firewalls and several routers¹ the desired outcome was to prevent any internal outbound connection from being able to connect directly to an Internet host, over any port, and vice versa. This meant installing a proxy server to handle all Internet requests. Our business requirement was to perform analysis and reporting on traffic levels, most visited sites, heavy users and busiest time periods, in addition to disallowing certain users from accessing any Internet location.



¹ Not all hardware is displayed here, only the most relevant to our study

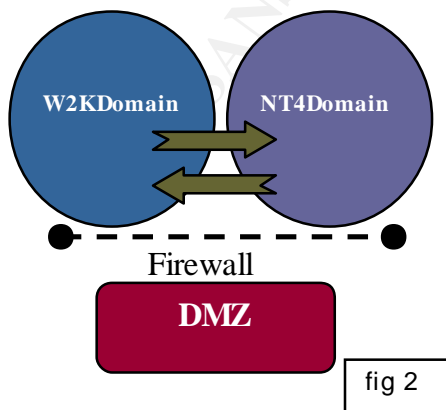
In order to monitor individual user's usage the proxy would be required to authenticate every request. We wanted the user experience to be as transparent as possible, users could previously gain seamless access to the internet via direct, unauthenticated connections to the internet, so it was important to minimise impact on their experience, prompting users for a username and password would not be good enough. So, our requirement was to implement a proxy service in our DMZ that could transparently authenticate user's network logins against one of our organisation's Windows 2000 Domain Controllers, through a highly secure internal firewall.

The Environment

Our organisation consists of one Windows 2000 Active Directory domain (Referred to as "W2Kdomain") and one NT 4 domain (Referred to as "NT4domain"), with a two-way trust operating between them. User workstations are predominantly Windows 95, with many 98, XP and Windows 2000 machines as well, and, as we found out later, a handful of iMac and G4 desktop machines. Table 1 lists some hosts that will be referenced in this paper and Figure 2 shows a logical layout of our Domains.

Description	Host Name	IP address
Internal Windows 2000 Domain Controller	W2KDC	192.168.1.2
DMZ Hosted Proxy Service	ISAPROXY	172.16.15.10
Internal Windows NT 4 Primary Domain Controller	NT4PDC	192.168.2.2

All desktop machines used some version of Internet Explorer for web access. Our chosen proxy software was Microsoft's ISA Server (This implies an easy solution to the problem, which is discussed in the later section, entitled "Why we didn't use ISA firewall"), installed on Windows 2000 Server. Sites are distributed across Australia, with many network subnets and a complex internal WAN configuration.



Why we didn't use the ISA firewall?

Given our topology of an internal and external firewall bastion the obvious solution would seem to be to use the ISA Server's inbuilt firewall functionality and utilise the ISA Server as the internal bastion. The internal interface of the firewall could communicate easily with our internal DC. Our own (healthy, I believe) level of scepticism prevented us from doing this. While our organization was happy to utilise Microsoft Products for the proxy service, their reputation in security had left us with little confidence in Microsoft's ability to produce a reliable firewall solution. Instead we chose a well-proven and trusted high-end product for our firewall solution.

Analysing the Proxy Server Behaviour

Our first step in configuring our internal firewall was to work out exactly what interaction the proxy server has with the internal DC. We built the Proxy server internally and joined the W2Kdomain. By chaining the new proxy to an ISP's proxy server we were able to run the server internally as a fully functional proxy. It forwarded all proxy traffic to the upstream server. Our main reason for doing this was to get the proxy functioning normally internally so we could examine the traffic that took place between the proxy and the DC whilst authenticating each request.

In order to examine TCP IP traffic we installed Netmon on the Proxy from the Windows 2000 CD. Netmon is a packet analyser included with the Windows 2000 package. Whilst not as functional as many sniffers available it has a couple of advantages; 1. It was included on the Windows CD, therefore no extra cost. 2. Easy to setup and proven reliability. Any thing that affects the TCP IP protocol stack has the ability to slow connections or cause instability on the system; therefore in this instance Netmon's simplicity was an advantage. We regularly run Netmon on production systems without any stability problems at all. Figure 3 shows the installation process on a Windows 2000 Server. What we saw was initially very worrying. We saw communication between the proxy and the site's DC over many ports, Kerberos, DNS, LDAP and RPC predominantly. Figure 4 shows Netmon reporting packets passed between the proxy server and the Domain Controller. On the screen are UDP packets associated with MS Kerberos, LDAP and MS RPC packets as well.

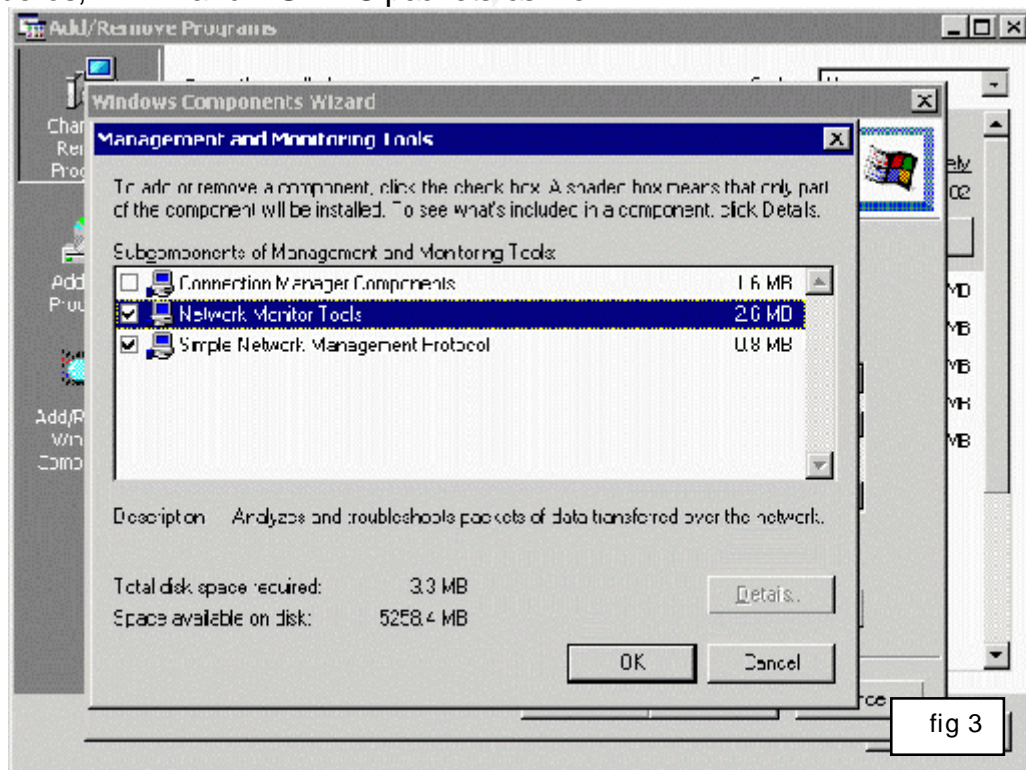


fig 3

The last entry in this table is the show stopper. The Remote Procedure Call protocol is essential in any networked environment, it is used in some form in both Windows and Unix systems (not always port 135 though) enabling one application to call a sub-routine from a service installed on another networked machine. Common uses are services such as NFS and the DCOM standard as well as Microsoft administrative functions. The behaviour of the RPC protocol makes limiting open ports very difficult. The operating system does not know what port RPC dependant services run on until they start up. After start up the port number is stored in the registry where the EndPointMapper can access it. When a client requests a service they first connect to the EndPointMapper on port 135, it reads the real port from the registry and sends the number back to the client. The client then disconnects and reconnects using the new port number². Accommodating this behaviour would require opening over 64000 ports for connections between the proxy and the internal DC, very bad.

For example, if an attacker were able to compromise the proxy server they could from there launch a wide range of RPC based attacks towards internal Domain Controllers. The “*Malformed RPC Request Can Cause Service Failure*”³ is just one of many RPC exploits that would be available to an attacker. As well as RPC other services operate in this range as well. Trojans such as Sub Seven are commonly set to use high ports⁴ and could be managed via a compromised DMZ host once they have been somehow deployed into the corporate network. The wide port range also facilitates port scanning, an attacker could gain valuable inside information regarding services running on the Domain Controller. Whilst our DMZ architecture offered other levels of protection against attack, such as our external firewall and router, making relaxed configuration decisions such as this goes against the preferred “defence in depth” approach. Our aim was to add layers of protection, not remove them. Implementing a firewall is an opportunity to control traffic passing through it, by opening this port range we would be essentially giving up this level of control and losing the ability to manage the traffic travelling between hosts. Unfortunately even restricting connections to only the two hosts; proxy and DC, does little to minimise the risk, as most attacks would target the DC anyway. Opening this port range facilitates many activities which firewalls are specifically designed to prevent. Anyone seriously considering opening this port range should probably not bother implementing a firewall at all.

Facilitating Communication

Steve Riley’s paper on “[Active Directory Replication over Firewalls](#)”⁵ offered three suggestions on how we could manage this broad range of traffic through our firewalls, each suggestion a gradual trade-off from flexibility and freedom to security. Whilst his article focuses mainly on facilitating Active Directory

² Riley, “Active Directory Replication over Firewalls”

³ MS, Microsoft Security Bulletin MS01-041

⁴ SANS, SubSeven Trojan v 1.1

⁵ See the bibliography for information on this excellent article.

replication, his solution also outlines strategies for dealing with the dynamic assignment of RPC port addresses which were relevant to our situation.

Our first option was of course to open up the required ports stated in table 2, including the dynamically assigned port range for RPC. As already noted, the advantage of not having to perform any configuration on either the DC or the proxy host is heavily outweighed by the compromise to firewall security. This option was never seriously considered.

The second option suggested using a registry hack to predefine the RPC ports used by Active Directory. By adding a new registry key it is possible to have the endpointmapper process always return the same port number, thus enabling us to define more specific firewall rules⁶. Whilst this option had some merit, and is certainly achievable, any registry hack carries with it some element of risk, and generally comes with limited support from vendors. Not so much risk of damaging the registry, but more a risk of unexpected behaviour. The implications of modifying registry keys is not always immediately apparent, and registry hacks always have an element of 'shoe horning' about them. In this situation a registry modification would be required to both proxy server and any DCs it needs to communicate with. It becomes a 'hidden' piece of configuration which could easily be lost after a rebuild or running a service pack. We decided against this approach mainly because the third option provided such a clean solution.

The third, and our chosen approach, was to take advantage of an interesting feature of Windows 2000 IPsec policies. Out of the box Windows 2000 is capable of encrypting almost all communication between Windows 2000 hosts using IPsec encryption. We discovered that by configuring our DC and proxy server to communicate only through IPsec we could restrict almost all our traffic to UDP port 500. IPsec implementations can be problematic, and are well known for their lack of vendor interoperability. However, in our situation we required only secure traffic within a homogenous environment of Windows 2000. The setup promised to be simple and straightforward.

IPsec can be applied in either AH or ESP mode. Authenticated Header (AH) is simply ensuring data integrity through the application of a MACing algorithm such as MD5. Data is unaltered and is transmitted in the normal fashion, over normal port ranges, this method does nothing to solve our problem. Encapsulated Security Payload (ESP) on the other hand, encrypts all data and packages it within an ESP packet⁷. By encapsulating traffic using IPsec we not only secure our traffic from possible eavesdroppers, we also get the added bonus of RPC traffic being restricted to UDP port 500. IPsec policies are easy to configure in Windows 2000. Many places detail how to create an IPsec policy, but our method contains one major difference from popular methods and therefore deserves further explanation. The steps we followed for creation of an IPsec policy are as follows,:

⁶ Riley, "Active Directory Replication over Firewalls"

⁷ Schmidt, p183

1. Open the Local Security Policy MMC (Administrative tools > Local Security Policy)
2. Right click 'IP Security Policies on Local Machine' and choose 'Manage IP filter lists and filter actions'.
3. The first tab is 'Manage IP Filter Lists', it allows for the creation of IP filter lists. Click 'add' and the wizard steps through the process of listing hosts, or address ranges that will be affected by the policy.
4. The second tab is 'Manage filter actions', it allows the definition of actions, in our case we can set the mode to ESP.
5. Once these elements are defined the policy is created simply by linking the filter list to the filter action together to form a rule, and defining the options for Authentication type, Tunnel setting and Connection Type.
6. Set the key exchange method.
7. Right click the new policy and select 'assign' to activate the policy. Unlike Group Policies, IPsec policies are active immediately.
8. Repeat the whole procedure on the Domain Controller.
9. Test the policy by performing a ping from one host to the other, a normal response proves the policy is working. A repeated message such as

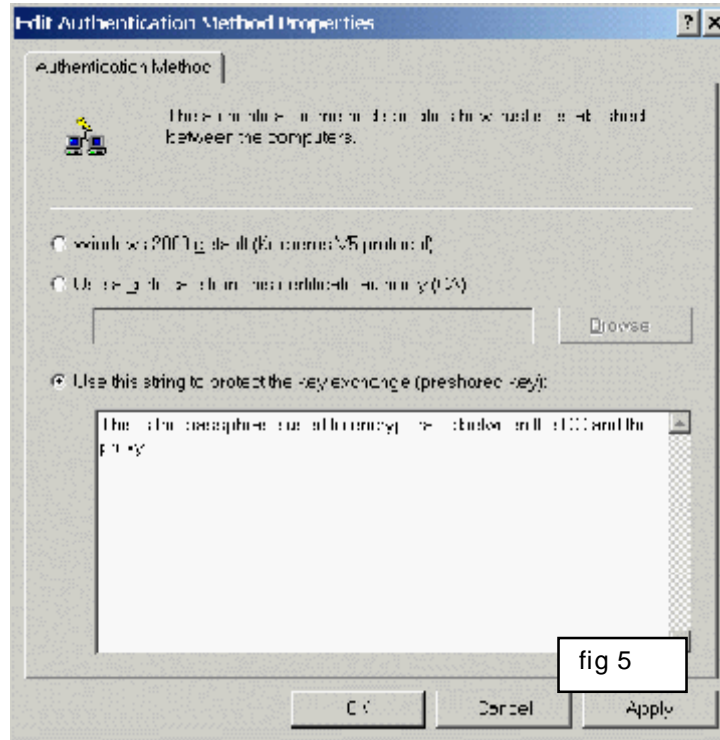
```
Negotiating security  
Negotiating security  
Negotiating security  
Negotiating security
```

indicated that the policy is not working and the hosts cannot exchange keys successfully, functional IPsec allows us to use ICMP where it would be normally blocked by the firewall, very handy. A more thorough test is to run IPSECMON, which provides reporting information on IPsec sessions, packets transferred and protocols utilised. Figure 6 is a sample screen capture from IPSECMON.EXE, it's installed by default with Windows 2000.

Regarding the use of pre shared keys; policies can use a variety of authentication methods to establish secure connections, including Kerberos, certificates and pre-shared keys. Whilst Kerberos offered higher security and least configuration, any problems with supporting services such as the W32Time service could potentially prevent the proxy from being able to authenticate the request, and therefore refuse the connection to the user. The time service has an inbuilt method for synchronising client/server clocks⁸, but in practice we found this to be happening far too often to be acceptable and we therefore found the most reliable method was to use a pre-shared key. Additionally Kerberos often took several seconds to exchange keys and begin secure transmission, whilst the pre-shared secret worked instantly. Figure 5 shows the setting of a pre-shared key. We could improve the security of this connection by utilising certificates on each host. However our organisation does not operate an internal certificate server, so these certificates would need to be purchased from, and renewed by, an external Certification Authority such as Thawte or Verisign. This adds an ongoing cost in terms of maintenance and purchasing of certificates.

⁸ MS KB 224799

Steve Riley strongly advises against the use of pre-shared keys in Windows only IPSec implementations, but in our case high security must be balanced with functionality, our main concern is in providing reliable and timely authentication. Using extra IPSec policies, as described later, gave us confidence that the proxy would be inaccessible to external attackers and that the key would remain secret.



© SANS Institute 2003

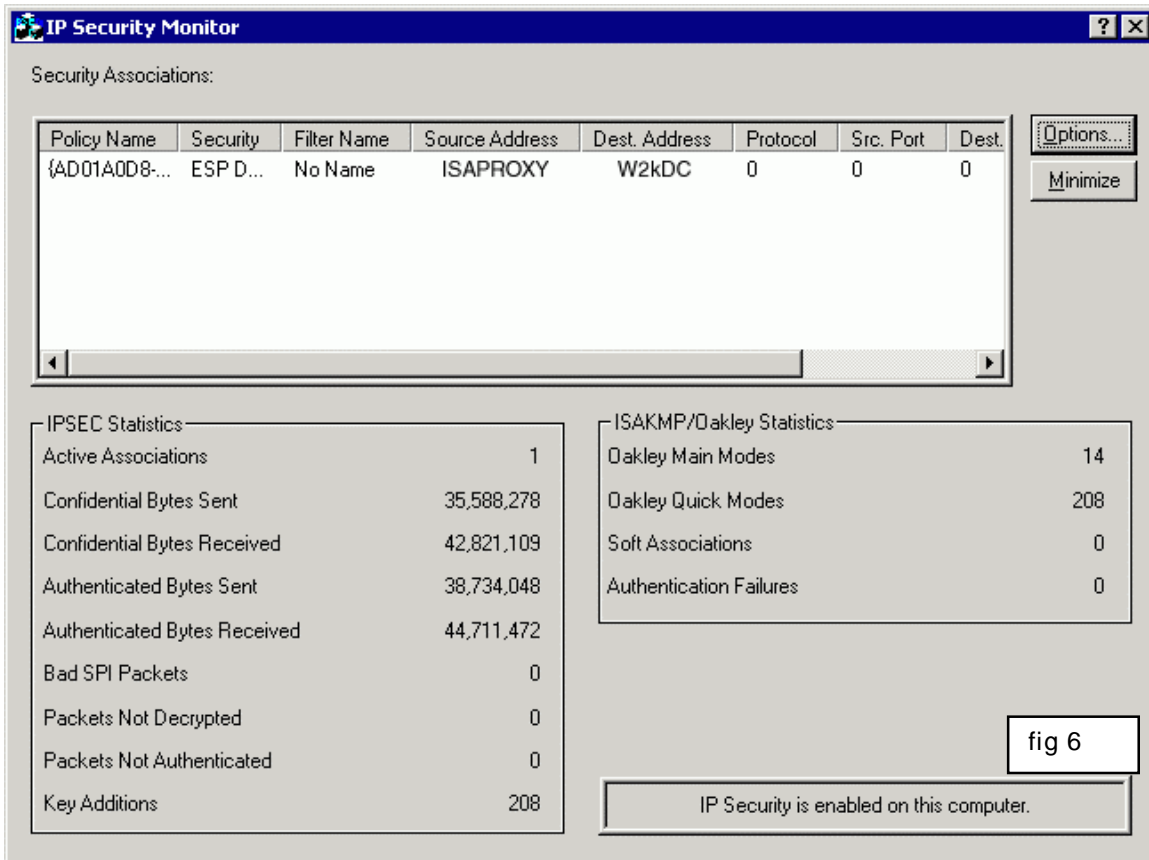


fig 6

IPSec protects all but broadcast, multicast, Kerberos, RSVP and ISAKMP (Internet Key Exchange) traffic. Therefore, based on Riley's recommendations, and our own findings, our final firewall rules looked like table 3.

table 3					
Source	Destination	Service	Port/Protocol	Action	
W2KDC	ISAPROXY	DNS	53/tcp 53/udp	allow	
W2KDC	ISAPROXY	Kerberos	88/tcp 88/udp	allow	
W2KDC	ISAPROXY	IPSec(includes ESP, IKE and AH)	500/udp	allow	
ISAPROXY	W2KDC	DNS	53/tcp 53/udp	allow	
ISAPROXY	W2KDC	Kerberos	88/tcp 88/udp	allow	
ISAPROXY	W2KDC	IPSec (includes ESP, IKE and AH)	500/udp	allow	

Configuring Name Resolution

Now that we could reliably communicate through the firewall we needed to configure name resolution to allow the proxy to resolve both internet and some internal host names.

Essential for the proxy server is the ability to resolve all internet URL requests from internal clients, therefore, DNS server entries in the TCP/IP configuration were set to internet DNS servers. Normally, with a large organisation, client machines utilise the closest DC to handle authentication, and by default Windows 2000 uses DNS to determine it's closest DC⁹. By utilizing external internet DNS servers in our proxies TCP/IP settings we had to find another way to firstly tell the proxy what server it should use as it's local DC, and secondly how to resolve it's name without using our internal DNS server. Microsoft Knowledge base article [Q180094](#) details how to write lmhosts files that predefine a DC host name using netbios. We added to the lmhosts file of the proxy the following lines:

```
192.168.1.2      W2KDC                #PRE #DOM:W2KDOMAIN
192.168.1.2      "W2KDOMAIN          \0x1b"             #PRE
```

We encountered a couple of curious features of this file's syntax. Firstly, all entries were required in uppercase. And secondly, the number of characters enclosed by quotation marks on the second line must equal 20¹⁰. With a native Windows 2000 domain this could be a problem for names longer than 15 characters, 5 characters are already used by the hex value at the end of the string. The #PRE flag indicates the entry should be preloaded in the netbios cache, preventing the machine from attempting to resolve the name externally. The second flag, #DOM, indicated that this address is the domain's DC address. The effectiveness of the file can be checked by running "nbtstat -R" to clear the cache, then "nbtstat -c" to display the cache contents. Both the DC hostname and the domain name should be listed.

We had things working, more or less, but needed to do some final detailed diagnosis to be confident our configuration was working. The Windows 2000 support tools provided on the Windows 2000 CD gave us detailed information on how well the proxy server was participating in the corporate domain. The first tool, netdiag.exe, logged detailed output on several network related functions, such as LDAP lookups, active IPsec policies and DNS queries. The results of this test were output to a log file, and are included as Appendix A. Of particular interest are the lines;

```
Trust relationship test. . . . . : Passed
  Secure channel for domain 'W2KDOMAIN' is to '\\W2KDC'.
```

confirming that the secure connection is functioning.

⁹ Microsoft, Windows 2000 DNS, p32

¹⁰ MS KB Q180094

Finally we ran nltest.exe which, when run from the client (the proxy server in this case), outputs a small report showing the clients current DC. The output;

```
C:\temp>nltest /dsgetdc:win2k
      DC: \\W2KDC
      Address: \\192.168.1.2
      Dom Guid: fd6f25a9-f5de-4060-8aa9-846dbe53c7ac
      Dom Name: W2KDOMAIN
      Forest Name: W2KDOMAIN.com
      Dc Site Name: CLOSESTSITE
      Flags: GC DS LDAP KDC TIMESERV WRITABLE DNS_FOREST
The command completed successfully
```

clearly indicates the proxy server regards W2KDC as it's closest DC and will therefore send authentication requests to this server.

Final Surprises

At this stage we had a fully functional proxy server capable of authenticating user credentials through the firewall and capable of resolving both internet URLs and internal DC host names. Upon moving the server into the DMZ, and changing the servers IP address to 172.16.15.10, we created acls in both W2kDomain and NT4domain and added all users we wished to grant internet access to. We named these acls "W2k_internet_access" and "NT4_internet_access" respectively. We then added both acls to the access control policy on the ISA server. What we found was that W2Kdomain users authenticated successfully and were granted transparent access to the internet. Unfortunately the NT4domain users could not authenticate at all. After examining the traffic leaving the proxy we found that it was attempting to authenticate NT4 users against the NT4 Primary Domain Controller, despite the server being a member of the W2KDomain. With NT4 having no support for IPSec we would not be able to configure a similar IPSec policy on NT4PDC. Luckily, nesting the NT4 acl in the Windows 2000 acl solved the problem, all authentication requests travelled via the Windows 2000 Domain Controller and the policy worked! All users could seamlessly access the internet using Internet Explorer, with the exception of the Mac users, who were required to use basic authentication¹¹. Figure 7 shows encrypted traffic passing between the proxy and the Windows 2000 Domain Controller, note the ESP protocol.

¹¹ Apple machines do not understand windows integrated authentication, even using Internet Explorer.

MAC Addr	Protocol	Description	Src Other Addr	Dst Other Addr	Type
L	ESP	SPI = 0x9534BA0E, Seq = 0x40	W2KDC	ISAPROXY	IP
L D62E19	ESP	SPI = 0x1336CBE, Seq = 0x3B	ISAPROXY	W2KDC	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x41	W2KDC	ISAPROXY	IP
L D62E19	ESP	SPI = 0x1336CBE, Seq = 0x3C	ISAPROXY	W2KDC	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x42	W2KDC	ISAPROXY	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x43	W2KDC	ISAPROXY	IP
L D62E19	ESP	SPI = 0x1336CBE, Seq = 0x3D	ISAPROXY	W2KDC	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x44	W2KDC	ISAPROXY	IP
L D62E19	ESP	SPI = 0x1336CBE, Seq = 0x3E	ISAPROXY	W2KDC	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x45	W2KDC	ISAPROXY	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x46	W2KDC	ISAPROXY	IP
L D62E19	ESP	SPI = 0x1336CBE, Seq = 0x3F	ISAPROXY	W2KDC	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x47	W2KDC	ISAPROXY	IP
L D62E19	ESP	SPI = 0x1336CBE, Seq = 0x40	ISAPROXY	W2KDC	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x48	W2KDC	ISAPROXY	IP
L D62E19	ESP	SPI = 0x1336CBE, Seq = 0x41	ISAPROXY	W2KDC	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x49	W2KDC	ISAPROXY	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x4A	W2KDC	ISAPROXY	IP
L D62E19	ESP	SPI = 0x1336CBE, Seq = 0x42	ISAPROXY	W2KDC	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x4B	W2KDC	ISAPROXY	IP
L D62E19	ESP	SPI = 0x1336CBE, Seq = 0x43	ISAPROXY	W2KDC	IP
L	ESP	SPI = 0x9534BA0E, Seq = 0x4C	W2KDC	ISAPROXY	IP

fig 7

Further use of IPsec to Harden the Proxy

Left in this state the proxy was essentially a corporate machine existing in the DMZ, so whilst we now had our required functionality we had also created an extremely attractive target for an attacker. We had managed our firewall configuration into a small set of rules, but, as with our first option discussed, we had now a DMZ based host capable of establishing connections to our internal Domain Controller. Every other host capable of this was situated behind two firewalls and several routers, making this machine considerably easier to get to. Our external firewall was configured to drop incoming connections to this host, but what if an exploit was found against our particular firewall product? Without a layered approach to security an attacker would have to overcome only one security hurdle (albeit a difficult one) to gain a foothold on into our organisation.

In order to close this security hole we would follow both the recommendations of our security experts and the instructions of another Steve Riley article, "[Using IPsec to Lock Down a Server](#)". Using the instructions in this article we modified our existing IPsec policy to refuse any connection originating from both our DMZ 172.16.15.0 network and the internal 192.168.0.0 networks, except on port 80, with the exclusion of the DC, of course. This behaviour was already configured into our firewalls, but now even in the event of either firewall failing to process requests properly, or the connection coming from another DMZ host, the proxy server would still deny any connection. Implementing this IPsec policy guarded against failure of either the internal or external firewall and added that extra layer of protection we wanted. Even if an attacker gained access to another DMZ server, they would be unable to find a route to the organisation's DC via the proxy.

Outcomes

By utilizing this unorthodox application of IPSec policies we have successfully achieved our goal of providing seamless, authenticated access to the internet through a high security firewall, using Microsoft's ISA Server in caching mode.

Over the last few months we've run this configuration in production with very few incidents, however, on occasions, even in this homogenous Windows 2000 server environment the IPSec connection has failed, causing the ISA service to hang or crash. We have managed these occasional incidents by utilising both ISA's inbuilt alert functionality and Windows 2000's ability to restart failed services automatically. We have kept a close eye on availability and decided that although not 100% reliable, the Windows IPSec implementation has suited our needs at a very low cost and with minimal maintenance cost. Our firewall rules were manageable and the addition of extra IPSec policies ensure the proxy is secure against external attacks.

© SANS Institute 2003, Author retains full rights.

References

- Microsoft, *2000 Windows 2000 Connectivity Through Firewalls*,
<http://support.microsoft.com/default.aspx?scid=kb;en-us;Q280132>
(6/08/2002)
- Microsoft Knowledgebase, *How to Write an LMHOSTS File for Domain Validation and Other Name Resolution Issues*,
<http://support.microsoft.com/default.aspx?scid=kb;en-us;q180094>
(6/08/2002)
- Microsoft Knowledgebase, *Basic Operation of the Windows Time Service* ,
<http://support.microsoft.com/default.aspx?scid=KB;en-us;q224799> (5/12/2002)
- Microsoft, *Malformed RPC Request Can Cause Service Failure*,
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-041.asp>
- Microsoft Whitepaper, *Windows 2000 DNS*,
<http://www.microsoft.com/windows2000/docs/w2kdns.doc> (1999)
- Riley , Steve, *Active Directory Replication over Firewalls*,
http://www.microsoft.com/serviceproviders/columns/config_ipsec_P63623.asp 6/08/2002
- Riley , Steve, *Using IPsec to Lockdown a Server*,
http://www.microsoft.com/serviceproviders/columns/using_ipsec.asp
6/08/2002
- SANS, *SubSeven Trojan v 1.1* , <http://www.sans.org/newlook/resources/IDFAQ/subseven.htm>
- Schmidt , Jeff, *Microsoft Windows 2000 Security Handbook*,
Que Publishers, 2000

© SANS Institute 2003. Author retains full rights.

Appendix A

Output from netdiag.exe command

```
Computer Name: ISAPROXY
DNS Host Name: isaproxy.w2kdomain.com
System info : Windows 2000 Server (Build 2195)
Processor : x86 Family 6 Model 8 Stepping 10,
GenuineIntel
List of installed hotfixes :
    Q147222
    Q295688
    Q320206
    Q321599

Netcard queries test . . . . . : Passed

Per interface results:

Adapter : primary

    Netcard queries test . . . : Passed

    Host Name . . . . . : isaproxy.w2kdomain.com
    IP Address . . . . . : 172.16.15.10
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : 172.16.1.1
    Primary WINS Server . . . . : 192.168.1.2
    Secondary WINS Server . . . :
    Dns Servers . . . . . : 139.130.4.4
                           139.134.2.190

    AutoConfiguration results . . . . . : Passed

    Default gateway test . . . : Passed

    NetBT name test . . . . . : Passed

    WINS service test . . . . . : Passed

Global results:

Domain membership test . . . . . : Passed

NetBT transports test . . . . . : Passed
List of NetBt transports currently configured:
```

```
NetBT_Tcpip_{C267FB17-901F-4C9B-BEE8-0CB82E526B3C}
1 NetBt transport currently configured.

Autonet address test . . . . . : Passed
IP loopback ping test. . . . . : Passed
Default gateway test . . . . . : Passed
NetBT name test. . . . . : Passed
Winsock test . . . . . : Passed
DNS test . . . . . : Passed
Redir and Browser test . . . . . : Passed
  List of NetBt transports currently bound to the Redir
  NetBT_Tcpip_{C267FB17-901F-4C9B-BEE8-0CB82E526B3C}
  The redir is bound to 1 NetBt transport.

  List of NetBt transports currently bound to the browser
  NetBT_Tcpip_{C267FB17-901F-4C9B-BEE8-0CB82E526B3C}
  The browser is bound to 1 NetBt transport.

DC discovery test. . . . . : Passed
DC list test . . . . . : Passed
Trust relationship test. . . . . : Passed
  Secure channel for domain 'W2KDOMAIN' is to '\\W2KDC'.

Kerberos test. . . . . : Passed
LDAP test. . . . . : Passed
Bindings test. . . . . : Passed
WAN configuration test . . . . . : Skipped
  No active remote access connections.

Modem diagnostics test . . . . . : Passed
IP Security test . . . . . : Passed
  Local IPSec Policy Active: 'ISA to DCs '

The command completed successfully
```



Upcoming SANS Training

[Click Here for a full list of all Upcoming SANS Events by Location](#)

SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS San Francisco Fall 2017	OnlineCAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced