



*Nail the Coffin Shut:*

**NTLM IS DEAD**

Kurt Grutzmacher - Defcon 16  
grutz @ jingojango.net



# Who I am...

Corporate Penetration Tester for nearly a decade (with a CISSP for business purposes, gotta do what ya gotta do)

Have seen the worst security get turned around into lil' better security

As the Enterprise learned how to protect themselves we had to figure out other ways to attack

This presentation is a culmination of this knowledge

Also dabble in Metasploit development, getting Free MacWorld passes, and spreading the good word of OWASP



# Quick Definitions

## LM - LAN Manager

Really old and really tired, never use this again

Finally disabled by default in Vista and Server 2008

## NTLM - NT LAN Manager

Replaced “LAN Manager” (for a good reason)

A “suite” of protocols for authentication and security: “NTLM Security Support Provider (NTLMSSP)”

Also known as “ntlm 0.12”

Describes an authentication protocol and the hash result

## Kerberos - Kerberos

But not *just* Kerberos, Microsoft extended Kerberos!



# Scrabble take two

Nonce - *Number Used Once*

Used to defeat replay attacks

SSPI - *Security Support Provider Interface*


Microsoft API to several security routines

SPNEGO - *Simple and Protected GSSAPI Negotiation Mechanism*

I don't know what to speak to you, so lets negotiate!

IWA - *Integrated Windows Authentication*

The act of negotiating authentication type using SPNEGO



# Windows "Type" Auth

NTLM Authentication Protocol is a challenge-response scheme that can be broken into three "Types":

*Type 1:* Client sends "Hi, I want to talk to you"

*Type 2:* Server sends "Ok, here's the various features and protocols I support including a *nonce* for you to encrypt your hashes with so nobody can replay it later in case they capture it. Oh and the domain you should authenticate to."

*Type 3:* Client response "Sweet, I agree on the features you desire and support them in my daily life. Here's the username, domain again, workstation name, and the encrypted LM and NTLM hashes."

The server recovers the LM/NTLM hashes and compares them to its internal table and grants / denies accordingly in the response to a Type 3 message.



# *LM is a rotted corpse*

1. Password converted to upper case
2. Password is null-padded or TRUNCATED to 14 bytes
3. Password is split into two halves of 7 bytes each
4. Two DES keys are created, one from each 7 byte half:
  - 4.1. Convert each half to a bit stream
  - 4.2. Insert a zero bit after every 7 bits
5. Each key DES-encrypts the string "KGS!@#\$\$%" creating two 8 byte ciphertext values
6. Concatenate the two results for your LM hash



# NTLM Protocol is...

1. Cleartext is converted to Unicode and hashed with MD4 -- This is the “NTLM Hash”
2. The 16-byte hash is null-padded to 21 bytes and split into three 7-byte values
3. These values are each used to create three DES keys
4. Each of these keys is used to DES-encrypt the nonce from the Type 2 message, resulting in three 8-byte ciphertext values
5. These three ciphertext values are concatenated to form a 24-byte value which goes into the Type 3 response.



# NTLMv2 Protocol is...


1. NTLM hash is generated
2. Unicode uppercase username and domain name are concatenated
3. An HMAC-MD5 of the NTLM hash and result from Step 2 is made
4. A blob is created using the timestamp, a client nonce and static data
5. An HMAC-MD5 of the blob and result from Step 3 is made
6. This 16-byte value result is used in the NTLM slot





# NTLMv2 Session...

1. An 8-byte client nonce is generated and padded to 24 bytes
2. The result is placed into the LM field of the Type 3 response -- No LM result is generated or passed using NTLMv2 Session
3. Server's nonce is concatenated with the client nonce -> Session nonce
4. Session nonce is MD5'd and truncated to 8 bytes -> Session hash
5. NTLM hash is generated, null padded to 21 bytes and split into three 7-byte values
6. These values are each used to create three DES keys
7. Each of these keys is used to DES-encrypt the nonce from the Type 2 message, resulting in three 8-byte ciphertext values
8. These three ciphertext values are concatenated to form a 24-byte value which goes into the Type 3 response.



Seems strong...

NTLM is better than LM:

1. Cleartext is NOT converted to upper case
2. Passwords are NOT broken into blocks of 7 bytes
3. DES not so good but it's the last step to generate results and client/server nonces protect from pre-computed attacks
4. Server nonces do *not* protect pre-computed attacks however.

In the end LM and NTLM hashes should be considered the same as cleartext. When obtained an attacker does not need to find the cleartext in order for them to be used.



# NTLM is supported...

...everywhere!

in Microsoft products (IIS, IAS, Exchange, Internet Explorer)

in Samba and Apache and PAM

in other browsers (Mozilla Firefox and Safari)

in proxy servers to support browsers who don't do NTLM

in your iPhone (really!) for Enterprises

in OSX to connect to Windows shares

in WinCE to connect to Windows shares

in ToasterBrandConsumerDevice to connect to Windows shares

in \* to connect to Windows shares



# So why is it dead?

NTLM has shown its survivability by hanging on to “backwards compatibility” and ubiquitous deployment. If it’s everywhere what is the incentive to get rid of it?

Good question, and for one-off sort of authentication the NTLM protocol is not a bad option. You’ve got:

- Replay protection

- Mixed case support from cleartext to ciphertext

- Client and Server nonces

- Message digests

- Timestamping



*..Sounds good so far!*

Lets not get ahead of ourselves just yet.

In an ENTERPRISE we have the joyful tune of “Single Sign-On”. When a workstation becomes a member of the domain any user that logs on can access their resources with only having to type their password once during the log on process

This means that the cleartext or LM/NTLM ciphertext may be stored within the memory of the workstation throughout the session or beyond!

It also means that authentication can happen at the request of an application and not by a user.



# Attack Scenarios



# Our Threat Model

The NTLM protocols pre-suppose an Enterprise authentication system using Windows Domains or Active Directory.

Evildoers must fit within this environment in order to take advantage of it so they usually have to physically have access inside.

Doesn't mean this isn't an external threat, just that at this time I can't think of or have seen an attack from the outside in.



# SMB Relay (original)

<http://www.xfocus.net/articles/200305/smbrelay.html>

First released in March, 2001 at @tlantaCon by Sir Dystic of cDc

Listens for NBT requests and collects LM/NTLM hashes for cracking

Version 1:

- Connects back to the requester using their credentials

- Emulates an SMB server for the attacker to connect to

- TCP/IP Addresses only

- Generally great for one-off attacks

Version 2:

- Supported NetBIOS names

- Relay to a third-party host





# SMB Relay (Metasploit)

Re-engineering of SMB Relay script as a Metasploit attack module

Metasploit already had LM/NTLM hash capture support since 2.7

Can connect back to original host or forward to a single host

Works **great** if:

- Users are local administrators

- Server service has been started on their workstations

- or the users have rights to your destination host

See last year's "Tactical Exploitation" presentation for other cool ideas.

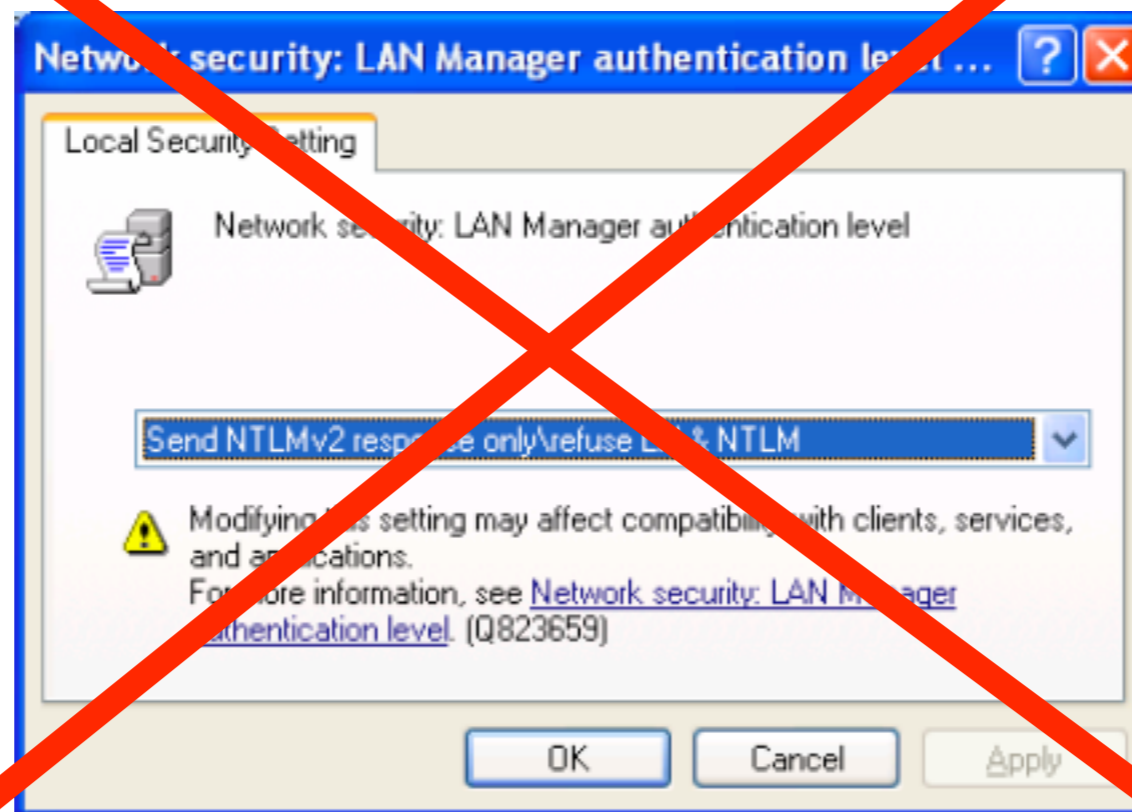
# Stopping SMBRelay

Through a GPO or within Local Security Policy change your LAN Manager authentication level:



# Stopping SMBRelay

Through a GPO or within Local Security Policy change your LAN Manager authentication level:



...but not really

NTLMv2 does not stop this attack. At the current time NTLMv2 is not *fully* supported within the MSF libraries so enable NTLMv2 (for now)





# Protocol Downgrade

During SPNEGO the client gets the first word on protocol support:

Signing, Sealing, use NTLM, Always Sign, send Target block, etc.

The server responds with their own list of support:

NTLM2 key, Target block included, 128-bit encryption, etc

If both sides agree the client sends all the requisite data for an authentication attempt and waits for a response.

Using MITM tools such as Cain & Able or Ettercap an attacker can force either side to negotiate LOWER than they would have otherwise.

# Protecting Downgrade

Through a GPO or within Local Security Policy change your LAN Manager authentication level:





# Replay Attacks

Comes in two forms:

- Network capture and replay if no nonce

- Obtaining the LM/NTLM hashes and using them during auth

“Pass The Hash” is the term and it is pure awesome:

- Obtain privileges on a server or workstation

- Dump a copy of stored hashes (SAM, LSASS, running processes)

- Skip the part of “converting to LM/NTLM” during the Network authentication routines

- Who needs to crack hashes anymore?



# Tools for Replay

Obtain hashes:

FGDump -

PWDumpX -

Cain & Able -

Pass The Hash Toolkit

Metasploit, Canvas, CORE Impact

Passing The Hash

Hydra

Pass The Hash Toolkit

Metasploit, Canvas, CORE Impact





# NTLM over ...

HTTP, IMAP, POP3, SMTP, NNTP, etc. . .

While NTLM is a Microsoft protocol, in order to fully support SSO it has to support standard protocols. NTLM “Type Messages” is the implementation of the NTLM Protocol over these 7-bit protocols.

Part of the Integrated Windows Authentication suite.

# IE Trust Zones

In order for Internet Explorer to perform Integrated Windows Authentication the browser must be in the “Local Intranet” or customized zone with unique security restrictions.





# *What to-do in a zone*

Perform automatic Integrated Windows Authorization

Instantiate more ActiveX/COM objects

Less restriction on existing ActiveX functions.



# Forcing Trust Zones

It has been possible in the past to force IE into the Local Intranet zone through the use of Flash or Java applets.

<http://heasman.blogspot.com/2008/06/stealing-password-hashes-with-java-and.html>



# Mozilla Auth Setup

In about:config

Enable IWA:

```
network.negotiate-auth.trusted-uris: list,of,uris  
network.negotiate-auth.using-native-gsslib: true
```

Or just NTLM:

```
network.automatic-ntlm-auth.trusted-uris: list,of,uris  
network.ntlm.send-lm-response: false
```



# NTLM in `<browser>`

Opera does not support NTLM authentication directly, you must go through a proxy server.

Safari for Windows will do NTLM but does not do Integrated Windows Auth. Typically a proxy server is used for OS X or stored credentials in the keychain.

Wget/CURL both support NTLM on the command line.

Links/Lynx ... why? maybe, not something I checked - usually use a proxy server like NTLMAPS.

# NTLM Message Header

Type 1 Message -> Client to Server

Bit	0								1								2								3								
Offset	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	
0	NTLMSSP\0								Type	Flags								Domain Buffer								Workstation Buffer							
32	OS Ver Structure								Workstation Data																								
64	Domain Data																																

# Flags

```
# NTLMSSP Message Flags
NEGOTIATE_UNICODE      = 0x00000001 # Only set if Type 1 contains it - this or oem, not both
NEGOTIATE_OEM          = 0x00000002 # Only set if Type 1 contains it - this or unicode, not both
REQUEST_TARGET         = 0x00000004 # If set in Type 1, must return domain or server
NEGOTIATE_SIGN         = 0x00000010 # Session signature required
NEGOTIATE_SEAL         = 0x00000020 # Session seal required
NEGOTIATE_LMKEY        = 0x00000080 # LM Session Key should be used for signing and sealing
NEGOTIATE_NTLM         = 0x00000200 # NTLM auth is supported
NEGOTIATE_ANONYMOUS    = 0x00000800 # Anonymous context used
NEGOTIATE_DOMAIN       = 0x00001000 # Sent in Type1, client gives domain info
NEGOTIATE_WORKSTATION  = 0x00002000 # Sent in Type1, client gives workstation info
NEGOTIATE_LOCAL_CALL   = 0x00004000 # Server and client are on same machine
NEGOTIATE_ALWAYS_SIGN  = 0x00008000 # Add signatures to packets
TARGET_TYPE_DOMAIN     = 0x00010000 # If REQUEST_TARGET, we're adding the domain name
TARGET_TYPE_SERVER     = 0x00020000 # If REQUEST_TARGET, we're adding the server name
TARGET_TYPE_SHARE      = 0x00040000 # Supposed to denote "a share" but for a webserver?
NEGOTIATE_NTLM2_KEY    = 0x00080000 # NTLMv2 Signature and Key exchanges
NEGOTIATE_TARGET_INFO  = 0x00800000 # Server set when sending Target Information Block
NEGOTIATE_128          = 0x20000000 # 128-bit encryption supported
NEGOTIATE_KEY_EXCH     = 0x40000000 # Client will supply encrypted master key in Session Key field of Type3 msg
NEGOTIATE_56           = 0x80000000 # 56-bit encryption supported
```



# NTLM Message Header

Type 2 Message -> Server to Client

Bit	0									1									2				3									
Offset	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
0	NTLMSSP\0									Type	Target Name									Flags				Nonce								
32	Context (Optional)									Target Information Security Buffer (Optional)									OS Version Structure (Optional)									...Data!				

# NTLM Message Header

*Target Information Security Buffer & Data*

TISB:

Length - 2 bytes

Space - 2 bytes

Offset - 4 bytes

Target Data:

Starts at TISB Offset

Type - 2 bytes

Length - 2 bytes

Data - Da Data

... Repeat ...

Terminator - (0x0000)

# NTLM Message Header

Type 3 Message -> Client to Server

Bit	1									2									3			
Offset	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
0	NTLMSSP\0								Type	LM/LMv2 Response						NTLM/NTLMv2 Response						Target ...
32	... Name			User Name						Workstation Name						Session Key optional			Flags			OS Version ... optional
64	OS Version ... optional																					

# HTTP NTLM Auth

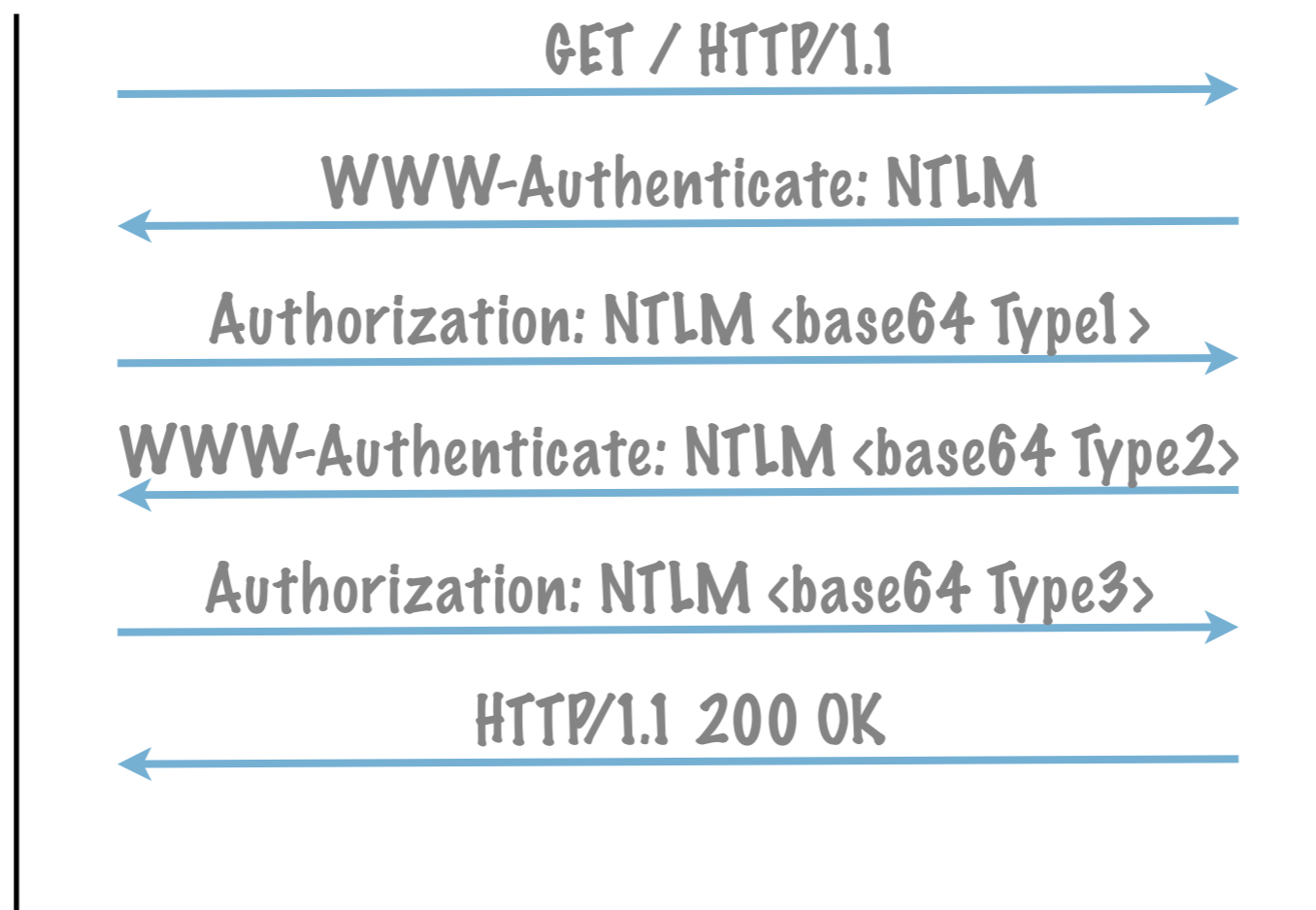
End User



Server



Session Established



# HTTP NTLM Auth

End User



Session Closed

Rogue Server



Session Established

Server



GET / HTTP/1.1

GET / HTTP/1.1

WWW-Authenticate: NTLM

WWW-Authenticate: NTLM

Authorization: NTLM <base64 Type1>

Authorization: NTLM <base64 Type1>

WWW-Authenticate: NTLM <base64 Type2>

WWW-Authenticate: NTLM <base64 Type2>

Authorization: NTLM <base64 Type3>

Authorization: NTLM <base64 Type3>

HTTP/1.1 200 OK

HTTP/1.1 200 OK



# What does that mean?

As a **Rogue Server** we're able to bridge authentication requests using Type Messages by directing HTTP requests to us (<img src>)

This was previously possible via SMBRelay but very limited target scope using WPAD+SOCKS or forcing file:// or smb:// connections

Jesse Burns @ iSEC Partners described the HTTP->SMB link in 2004 but never released source code

In late 2007 I implemented a hash collector and HTTP->IMAP bridge

This year Eric Rachner released "scurvy"

So what's new?



# Introducing Squirtle



# Squirtle? What the...

Squirtle is a **Rogue Server** with **Controlling Desires!** It does not require:

Man In The Middle techniques such as:

- ARP Poisoning

- DNS Redirection

- GRE Tunnels

Squirtle does require:

- The browser be in a “trusted zone” for IWA to work

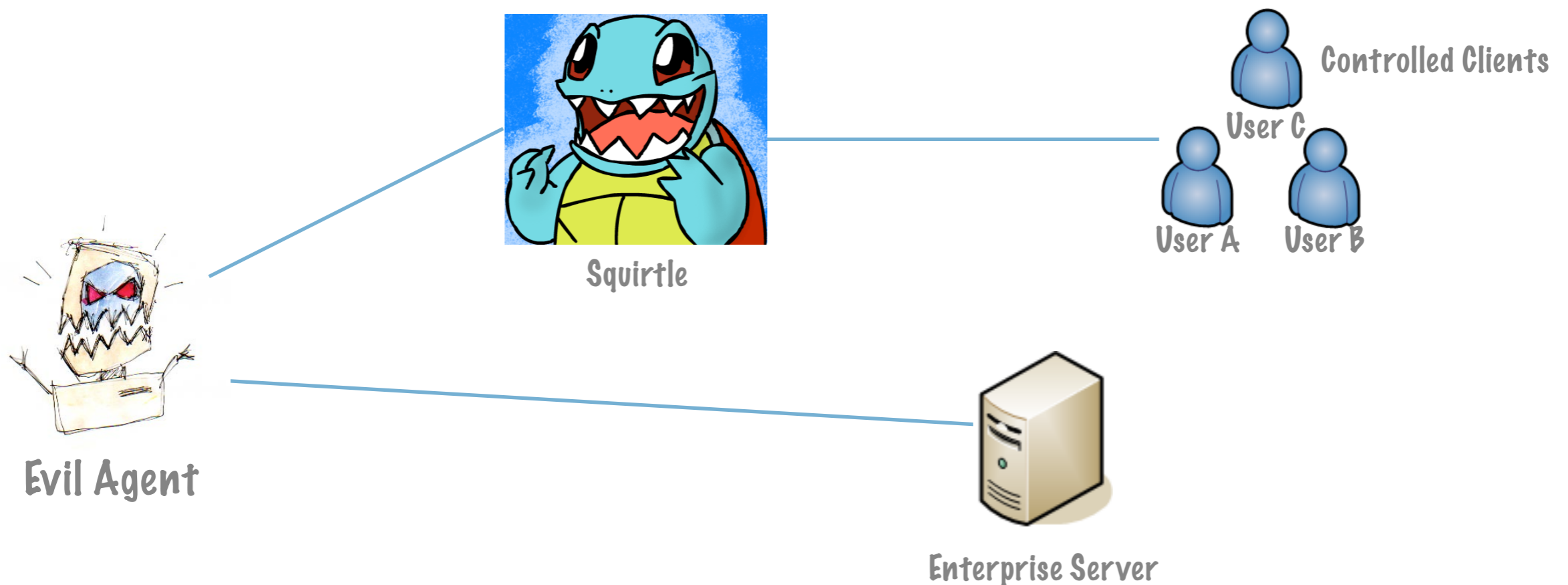
- Support for WWW-Authenticate: NTLM


- You somehow direct the browsers to it (XSS, proxy, <img>, etc)



# What... does it do?

Squirtle listens and collects web clients to use during IWA requests. Any external agent can use the provided API to request authentication from specific users using a given nonce from an enterprise server.



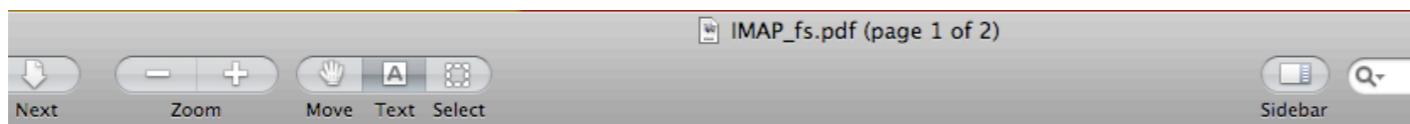


# What does this mean?

Past attacks against Windows authentication have been either directed at a single server or back towards the client.

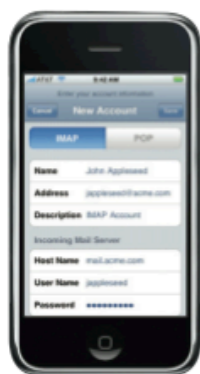
By corralling clients and exposing an API to externally written tools, Squirtle allows proxy servers to be written in any language that the attacker desires. They don't need to worry about grabbing clients and holding on to them, let Squirtle do that.

Existing frameworks such as Metasploit, Canvas and CORE Impact can use Squirtle to perform attacks against resources that require authentication without having to obtain cleartext or LM/NTLM hashes!



## iPhone and IMAP

iPhone also supports strong authentication methods, including industry-standard MD5 Challenge-Response and **NTLMv2**.



**IMAP or POP-enabled mail solutions**  
iPhone supports industry-standard IMAP4- and POP3-enabled mail solutions on a range of server platforms, including Windows, UNIX, Linux, and Mac OS X.

Additional information regarding the IMAP4rev1 standard can be found at [www.imap.org](http://www.imap.org).

With support for the IMAP mail protocol, iPhone can integrate with just about any mail server environment. If the server supports IMAP and is configured to require user authentication and SSL, iPhone provides a highly secure, standards-based approach to email deployment. In a typical deployment, iPhone establishes direct access to an IMAP-enabled server over port 993 and access to SMTP servers over port 587. These servers can be located within a DMZ subnetwork, behind a corporate firewall, or both. With SSL, iPhone supports 128-bit encryption and X.509 root certificates issued by the major certificate authorities. iPhone also supports strong authentication methods, including industry-standard MD5 Challenge-Response and NTLMv2.

### IMAP Network Setup

The IT or network administrator will need to complete these key steps to enable direct access from iPhone to an IMAP-enabled mail solution:

- Open port 993 to allow email to be received through the firewall. The proxy server must be set to IMAP over SSL. SSL ensures that mail is securely encrypted during wireless transmission.
- As a best practice and for additional security protection, install a digital certificate on the server from a trusted certificate authority (CA) such as VeriSign. Installing a certificate from a CA is an important step in ensuring that your proxy server is a trusted entity within your corporate infrastructure.



# Thinking about it...

I came up with this particular scenario and tool because after finding a ton of internal XSS vulnerabilities the response was “Great, you can run a port scanner or send print jobs. What else?”

So think about this:

- Internal servers with web programming errors (XSS, SQL, etc)

- Open SMB c:\inetpub\www shares with write access

- An internal PHPNuke instance

- Sending an E-mail with a link inside of it

- The evil act of opening Microsoft Office documents

They will all be controlled by the mighty Squirtle!



# Core Functions

Clients are given a pre-computed key during their first connection as a Cookie. A keepalive is sent to the Squirtle controller for action requests.

## Agent Functions

/controller/listclients -- List collected clients

/controller/listhashes -- List collected hashes and nonces

/controller/static -- Request client auth with static nonce

/controller/type2 -- Request client auth with a given nonce

/controller/redirect -- Force a redirect and drop the client

All functions require Basic Auth be supported to keep the riff-raff out.

## Client Functions

/keepalive -- Hi, i'm still here.. Got anything for me?

/client/auth -- Oh, ok I'll go here.. Authenticate? Maybe...

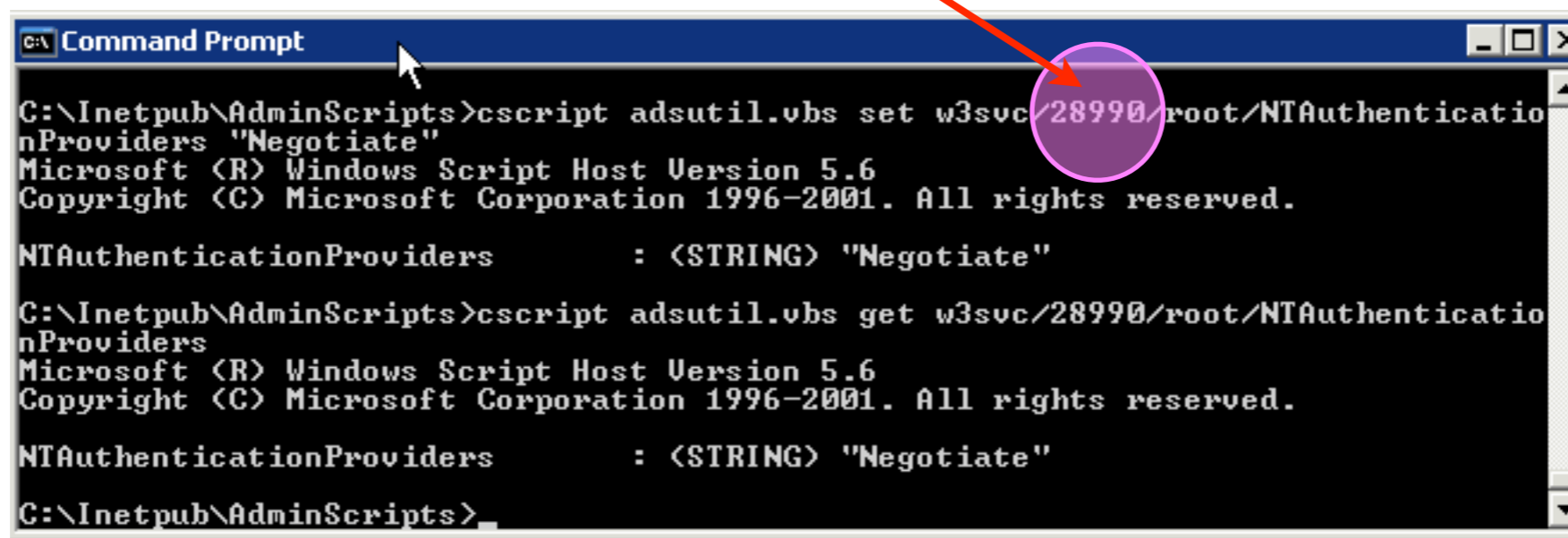


Demos

# Protecting IIS

For each instance, follow <http://support.microsoft.com/kb/215383>

**cscript adsutil.vbs set w3svc/instance#/root/NTAuthenticationProviders "Negotiate"**



```
C:\Inetpub\AdminScripts>cscript adsutil.vbs set w3svc/28990/root/NTAuthenticationProviders "Negotiate"
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

NTAuthenticationProviders      : <STRING> "Negotiate"

C:\Inetpub\AdminScripts>cscript adsutil.vbs get w3svc/28990/root/NTAuthenticationProviders
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

NTAuthenticationProviders      : <STRING> "Negotiate"

C:\Inetpub\AdminScripts>
```

This will break NTLM-only supported systems like NTLM proxies so do your testing before hand.




# Forcing the client

Not possible. If a browser is in the Local Intranet zone and sees the “WWW-Authenticate: NTLM” header they will attempt to authorize with it.

Best bet at the moment is to enable NTLMv2-only and get rid of all your Windows NT servers (you have gotten rid of them, right? RIGHT?)

At least with NTLMv2 decryption will take a long long long time should an attacker obtain user authentication packets.





# Q&A - URLs

*Squirtle:* <http://code.google.com/p/squirtle>

*Pass The Hash Toolkit:* <http://oss.coresecurity.com/projects/pshtoolkit.htm>

*FGDump:* <http://www.foofus.net/fizzgig/fgdump/>

*Cain & Abel:* <http://www.oxid.it/cain.html>

<http://grutztopia.jingojango.net/> -- <http://www.metasploit.com/>