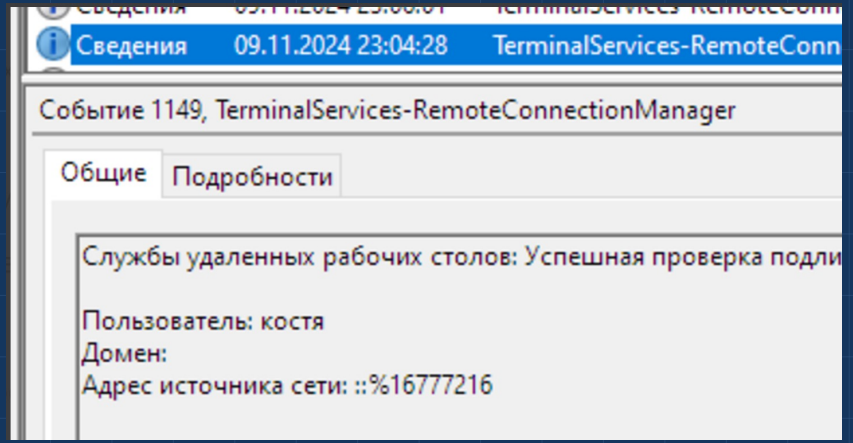
– SLIDE 00 / TITLE –

::%16777216

артефакт ngrok или баг в Windows?



Константин Грищенко · Positive Technologies · 26.05.2026

– SLIDE 01 / whoami –

Константин Грищенко

отвечаю за развитие технологий SOC в PT X

более 5 лет в SOC на различных должностях

23 года в практической ИБ

немного младше протокола IPv4

сильно старше чем IPv6

однажды смог установить Win95 с кучи 3½” дискет

Примеры упоминаний

- **Finding** > "C:\Windows\system32\ngrok.exe" tcp 3389
  - **Static variables** > **ngrok | tcp 3889**
  - **Query** > Sophos_process_journal
    - CMDLine
      - ngrok
      - tcp 3389
- **Windows Event Logs** (Microsoft-Windows-TerminalServices-RemoteConnectionManager%4Operational.evtx)
  - **Finding** > Source Network Address: ::%16777216
    - **Static variable** > **::%16777216**
    - **Query** > Rapid Response: Logins.01.0 – 1149 RDP Logins
      - Source IP
        - %::%16777216%

https://news.sophos.com/en-us/2022/07/14/rapid-response-the-ngrok-incident-guide/

Hunting and remediating ngrok tunnels using Logpoint

May 11th, 2022 - 4 min read

by Bhabesh Raj Rai, Security Research

Among developers, ngrok is a popular reverse proxy utility for exposing internal services to the internet by routing traffic through its cloud network. Inspired by a Twitter thread, let's walk through the process of detecting ngrok's remote desktop protocol (RDP) tunnel.

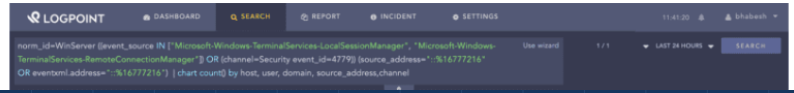
ngrok is very easy to use and does not rely on port forwarding, dynamic DNS or VPN. Naturally, what makes
persist

In 201
report
MAZE
range

Detecting ngrok with Logpoint

Creating an RDP tunnel by ngrok leaves a source address value of **::%16777216** in the system's RDP event logs. Defenders can use this artifact to hunt down reverse RDP tunnels in their environment.

```
soft-Windows-TerminalServices-LocalSessionManager",
alServices-RemoteConnectionManager"])
_id=4779)) (source_address="::%16777216" OR eventxml.address="::%16777216")
```



https://www.logpoint.com/en/blog/a-deep-look-at-the-darkside-ransomware-operators-and-their-affiliates/

Еще примеры упоминаний

Итоги проектов по расследованию инцидентов и ретроспективному анализу — 2023–2024

Содержание:

чаще всего по протоколу RDP) к внутренним ресурсам инфраструктуры заказчика. При использовании **Ngrok** нет возможности определить IP-адрес узла атакующего, поскольку сервис использует собственную сетевую инфраструктуру для организации подключения, что дает злоумышленникам дополнительное преимущество. На факт использования **Ngrok** может указывать наличие значения ::%16777216 в поле IP-адреса источника журналов подключения (ОС Windows) на конечном узле (в качестве дополнительного признака; само по себе значение ::%16777216 может фигурировать в журналах и при использовании иных решений, например RDG).

https://ptsecurity.com/research/analytics/itogi-proektov-po-rassledovaniyu-inczidentov-i-retrospektivnomu-analizu-2023-2024/#id18

И еще примеры упоминаний

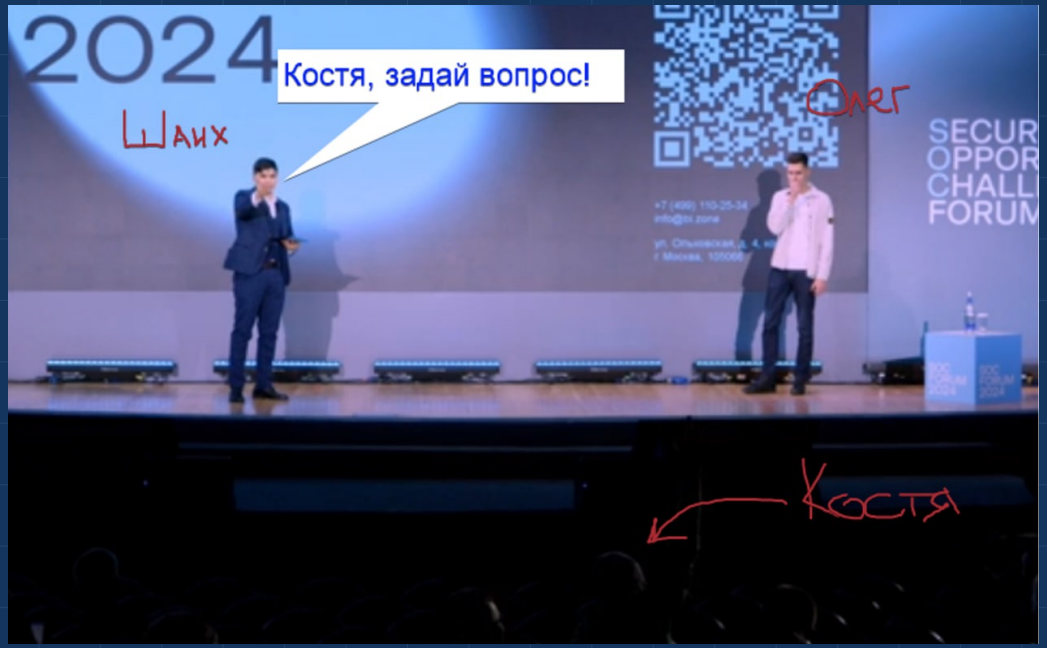
Злоумышленники прибегают к исполь
Control, Protocol Tunneling, T1572) или
Access Software, T1219) по нескольким

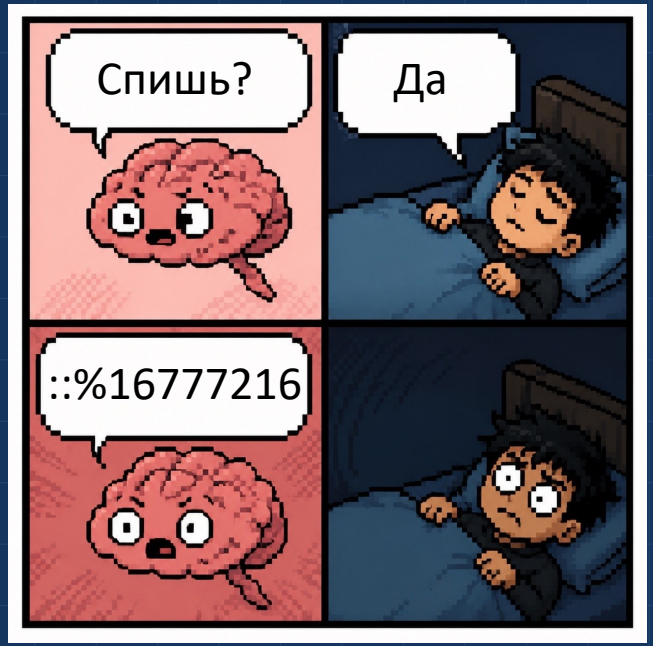
Во-первых, это позволяет обойтись б
войти в систему в целевой инфраструк
подрядчика. При этом зачастую атаку
настроенном мониторинге подключен
адресов (например, принадлежащих п
сигналом тревоги для сотрудников слу
зафиксирована, то, скорее всего, посл
как следствие, злоумышленники потеряют доступ к инфраструктуре.

https://securelist.ru/trusted-relationship-attack/109620/

Так, например, при обычном подключении по протоколу RDP в журнале Microsoft-Windows-TerminalServices-LocalSessionManager/Operational.evtx мы увидим события подключения (ID 21) или переподключения (ID 25), где в поле источника подключения будет указан IP-адрес злоумышленника (внешний IP-адрес, если система доступна из интернета, или внутренний IP-адрес другой скомпрометированной системы). Если подключение по RDP выполнено через утилиту для туннелирования, в журнале в качестве источника подключения будет указано значение *::%16777216* — оно не несет никакой информации о подключившейся системе. В большинстве случаев такой артефакт будет лишь признаком подключения через утилиту туннелирования.

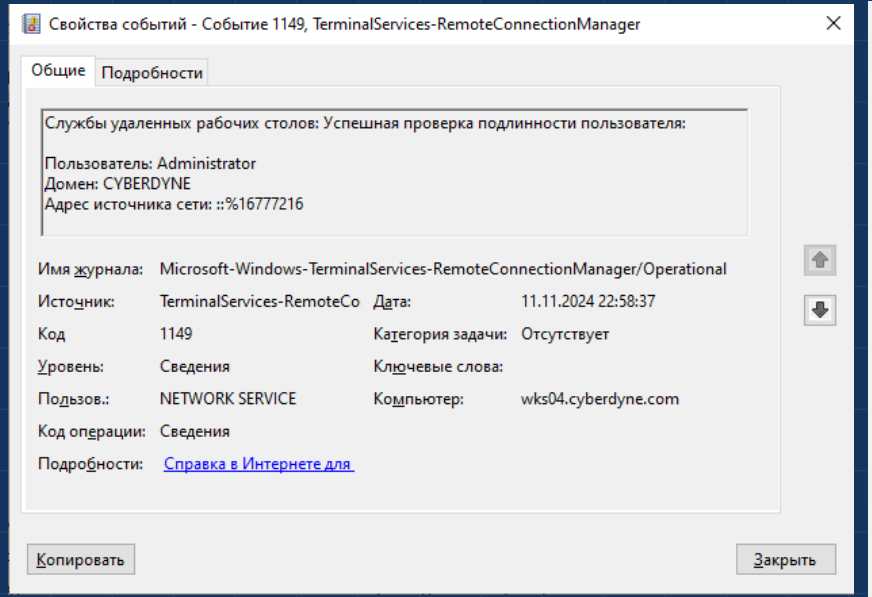
Зачем ходить на конференции?

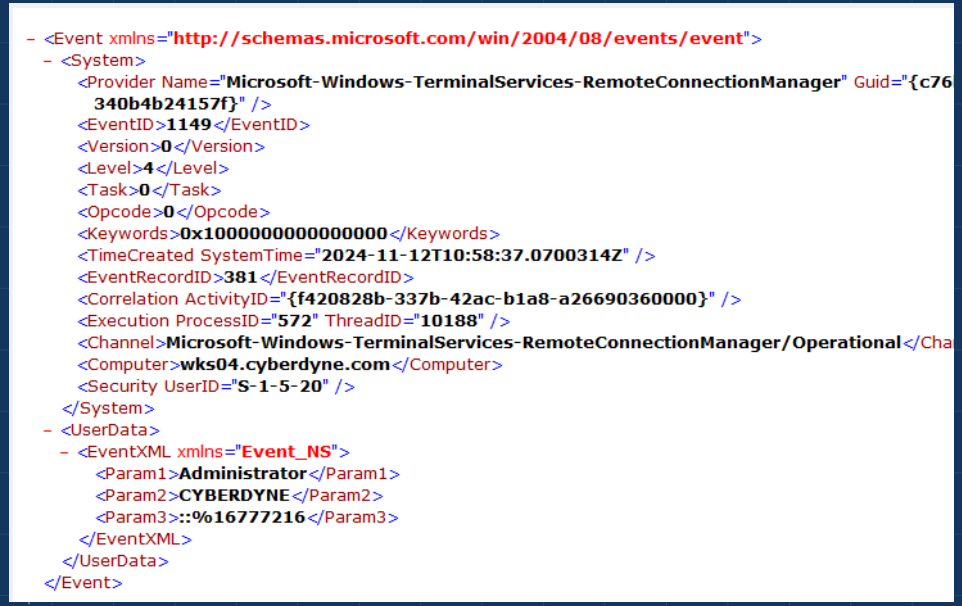




– SLIDE 06 / 41 –

EVENT ID 1149





Журналы приложений и служб > Microsoft > Windows > TerminalServices-RemoteConnectionManager > Operational

Какие бывают IP-адреса?

IPv4 32 бита

Internet address value	Meaning
"4.3.2.16"	Decimal
"004.003.002.020"	Octal
"0x4.0x3.0x2.0x10"	Hexadecimal
"4.003.002.0x10"	Mix

```
PS C:\Users\user> ping 1539747520

Обмен пакетами с 91.198.174.192 по с 32 байтами данных:
Control-C
```

https://datatracker.ietf.org/doc/html/rfc791
https://ru.manpages.org/inet_aton/3
https://learn.microsoft.com/ru-ru/windows/win32/api/wsipv6ok/nf-wsipv6ok-inet_addr

– SLIDE 08 / 41 –

Какие бывают IP-адреса?

IPv4 32 бита

Internet address value	Meaning
"4.3.2.16"	Decimal
"004.003.002.020"	Octal
"0x4.0x3.0x2.0x10"	Hexadecimal
"4.003.002.0x10"	Mix

```
PS C:\Users\user> ping 1539747520

Обмен пакетами с 91.198.174.192 по с 32 байтами данных:
Control-C
```

https://datatracker.ietf.org/doc/html/rfc791
https://ru.manpages.org/inet_aton/3
https://learn.microsoft.com/ru-ru/windows/win32/api/wsipv6ok/nf-wsipv6ok-inet_addr

IPv6 128 бит

Предпочтительная форма — x:x:x:x:x:x:x:x, x — шестнадцатеричные значения восьми 16-разрядных фрагментов адреса.

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
1080:0:0:0:8:8:800:200C:417A

Доступен специальный синтаксис для сжатия нулей:
FF01:0:0:0:0:0:0:43
FF01::43

```
c:\Users>ping 2620-0-862-ed1a--1.ipv6-literal.net

Pinging 2620:0:862:ed1a::1 with 32 bytes of data:
```

https://datatracker.ietf.org/doc/html/rfc4291
https://datatracker.ietf.org/doc/html/rfc1924

Какие бывают IP-адреса?

IPv6 128 бит

Могут там встречаться символы `%`?

Могут:

Scoped literal IPv6 addresses (with zone index)

- fe80::1ff:fe23:4567:890a**%3**
- fe80::1ff:fe23:4567:890a**%eth2**

https://datatracker.ietf.org/doc/html/rfc4007

Нет ничего похожего на это

::%16777216

Пора выдвигать гипотезу и проверять

Нет ничего похожего на это

::%16777216

Но на самом деле есть

$16\,777\,216_{dec} = 01\ 00\ 00\ 00_{hex}$
т.е. одна единица и много нулей

::1 = 0:0:0:0:0:0:0:1
адрес localhost в IPv6 – это тоже одна единица и много нулей

Гипотеза:
::%16777216 в логе – это «странная» форма записи IPv6 адреса ::1

Уточним немного гипотезу

::%16777216 в логе – это «странная» форма записи IPv6 адреса ::1

Откуда берётся «странность»?

- виноват ngrok?
- виноват не ngrok?

Проще в проверке гипотеза:

::%16777216 в логе – это «странная» форма записи IPv6 адреса ::1, возникающая не из-за ngrok

Как проверять?

Взять что-нибудь другое и провести эксперимент

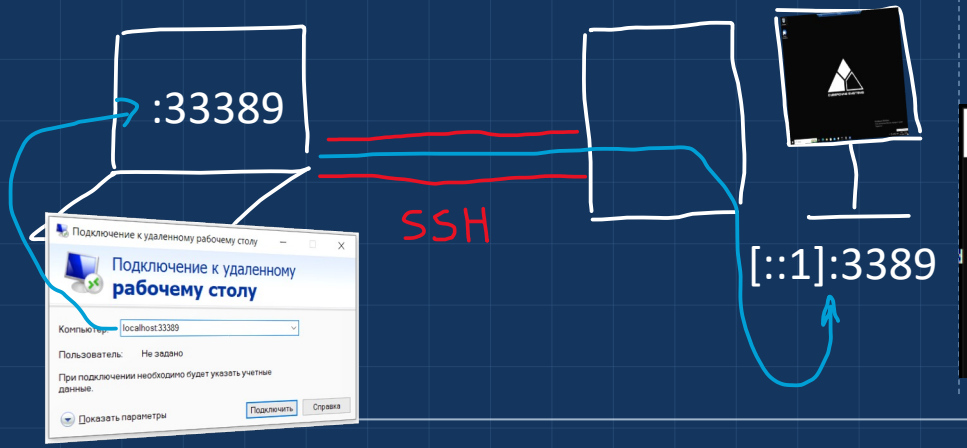
https://github.com/anderspitman/awesome-tunneling

– SLIDE 12 / 41 –

Ээээксперименты!

ngrok? localtunnel? …

```
ssh -L 33389:[::1]:3389 pc.testlab.lan
```



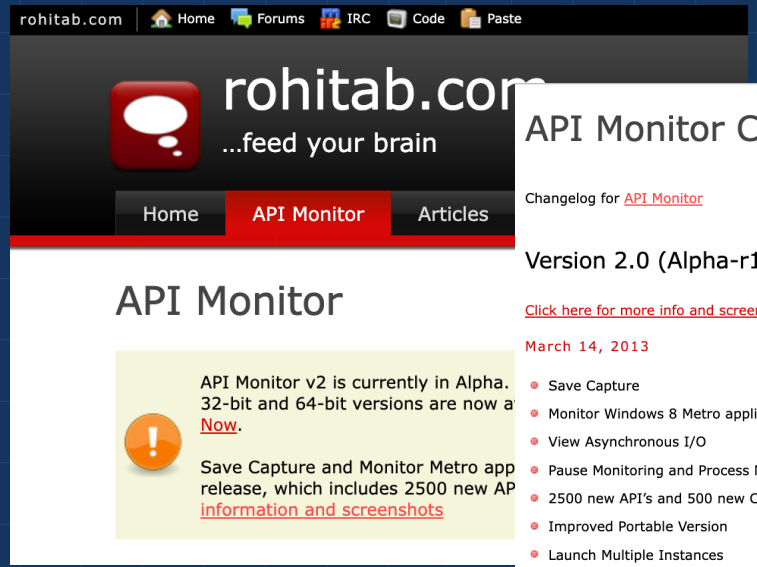
Администратор: Командная строка

```
 PolicyAgent
[svchost.exe]
 TCP    [::1]:3389             [::1]:57888            ESTABLISHED
 TermService
[svchost.exe]
 TCP    [::1]:57888            [::1]:3389             ESTABLISHED
[sshd.exe]
 UDP    0.0.0.0:123            *:*
```

Пара слов про API Monitor

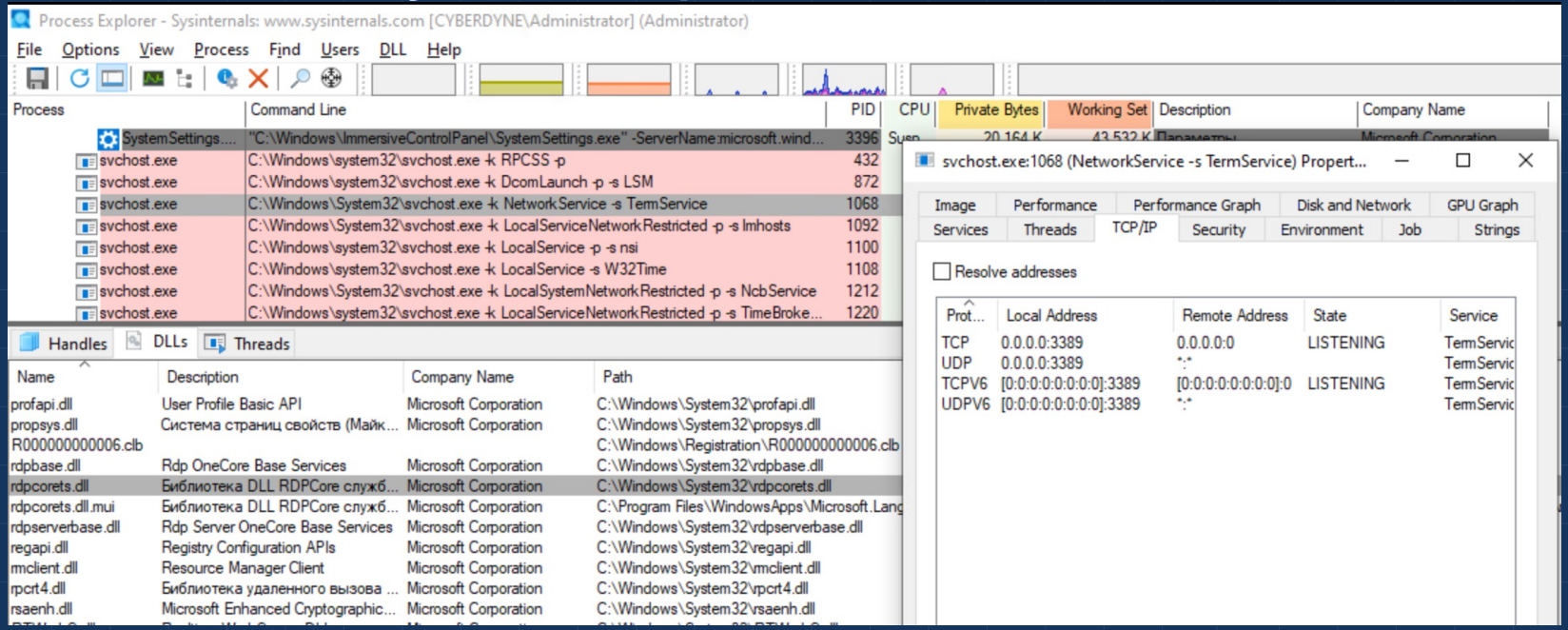
API Monitor is a free software that lets you monitor and control API calls made by applications and services.

Its a powerful tool for seeing how applications and services work or for tracking down problems that you have in your own applications.

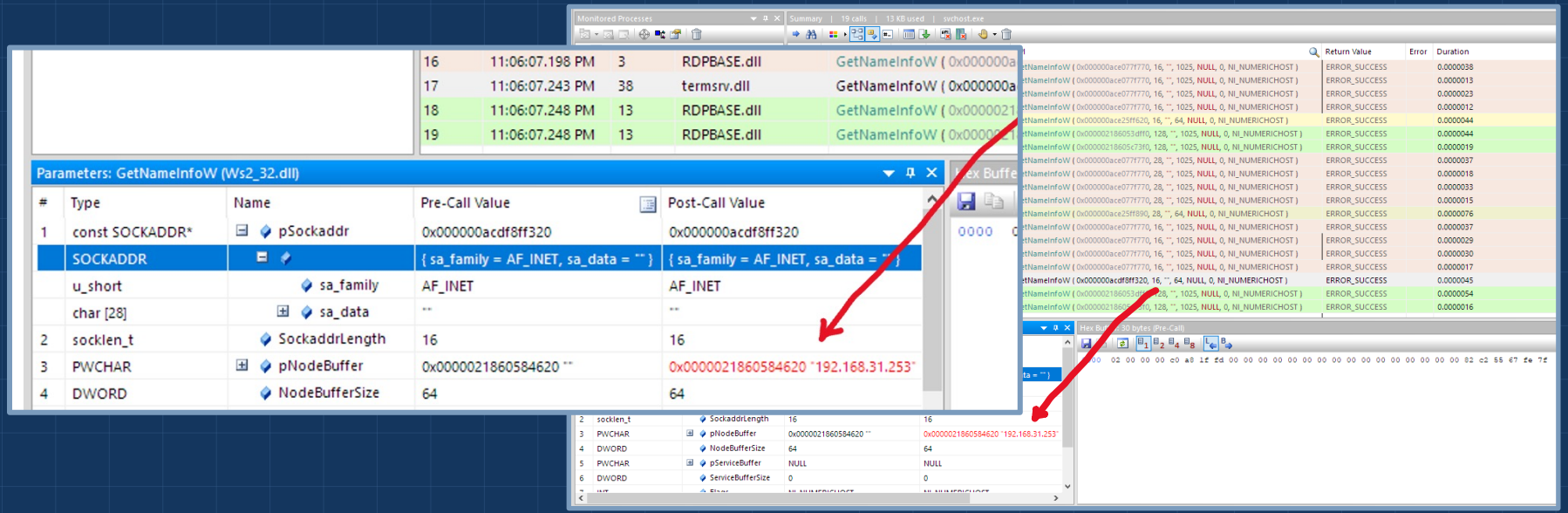


http://www.rohitab.com/apimonitor

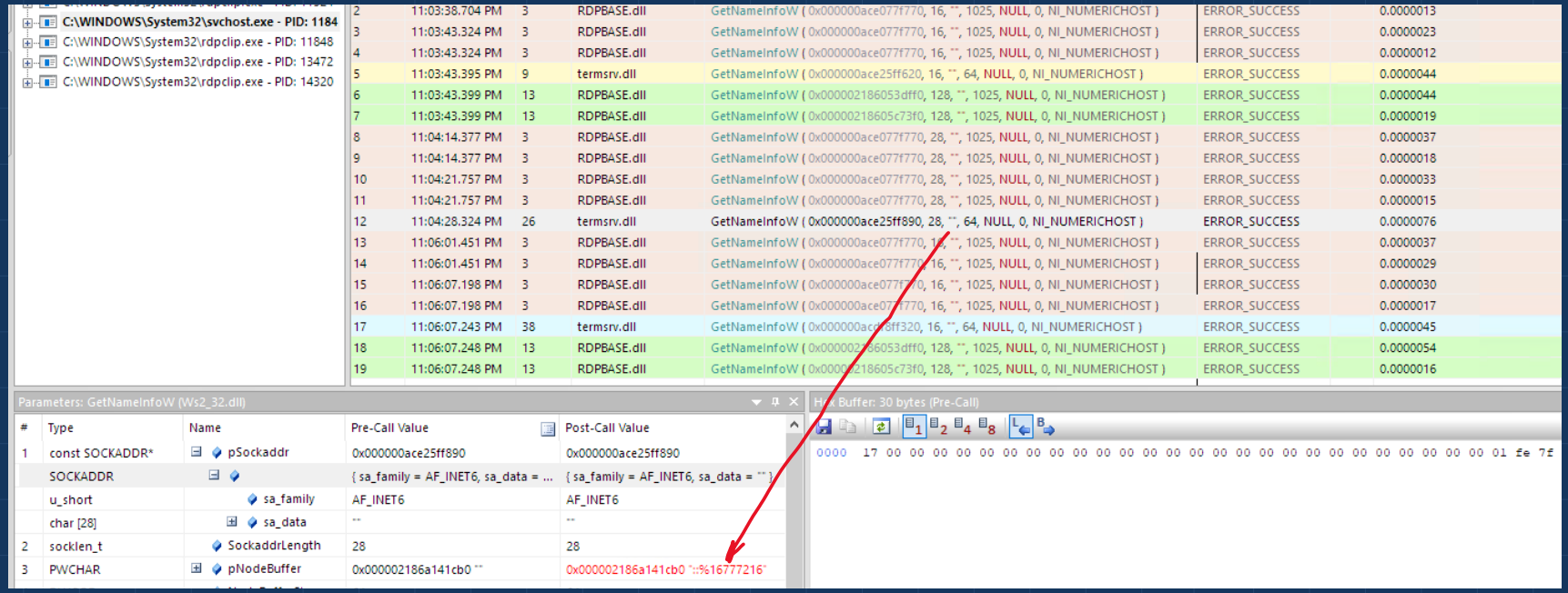
Как найти нужный процесс?



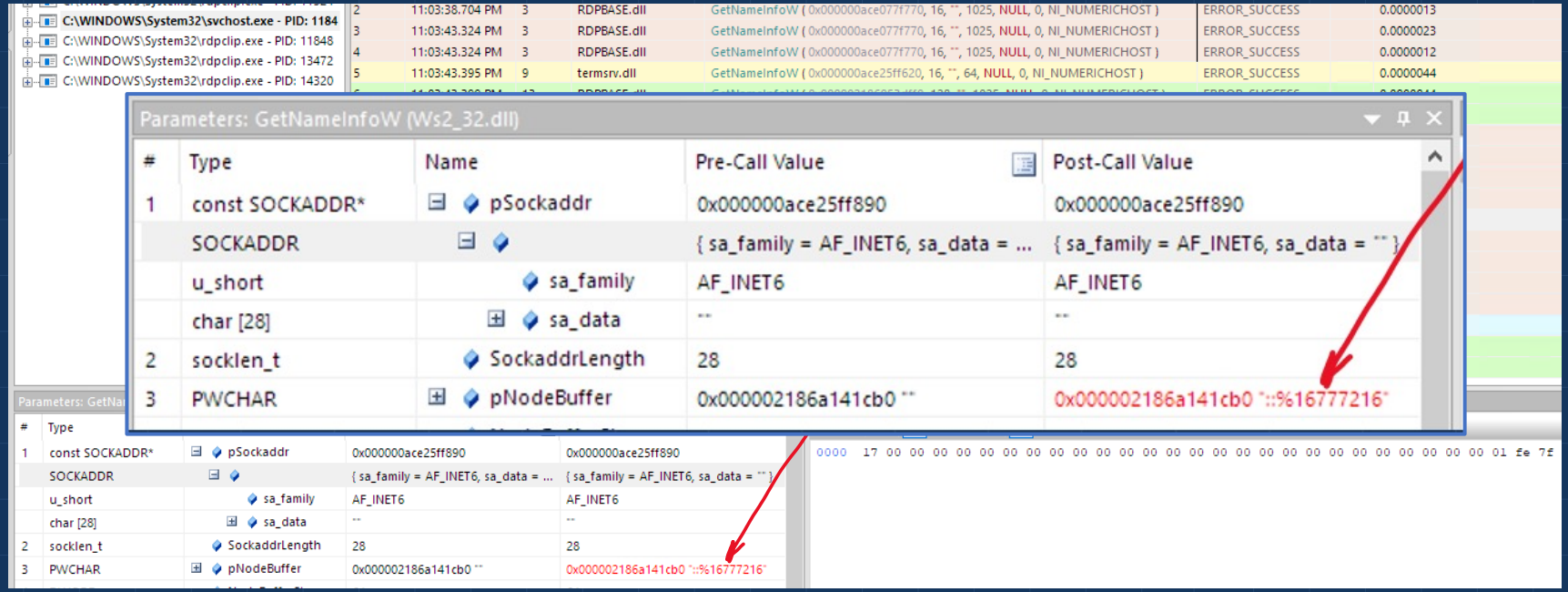
Расследование



Расследование



Расследование



GetNameInfoW

C++

```
INT WSAAPI GetNameInfoW(
  [in]  const SOCKADDR *pSockaddr,
  [in]  socklen_t      SockaddrLength,
  [out] PWCHAR         pNodeBuffer,
  [in]  DWORD          NodeBufferSize,
  [out] PWCHAR         pServiceBuffer,
  [in]  DWORD          ServiceBufferSize,
  [in]  INT            Flags
);
```

NI_NUMERICHOST - возвращает числовую форму имени узла вместо его имени

```
GetNameInfoW ( 0x000000b47d87f530, 28, "", 64, NULL, 0, NI_NUMERICHOST )
```

```
GetNameInfoW ( 0x000000acdf8ff320, 16, "", 64, NULL, 0, NI_NUMERICHOST )
```

https://learn.microsoft.com/ru-ru/windows/win32/api/ws2tcpip/nf-ws2tcpip-getnameinfow

SOCKADDR

IPv4 sockaddr_in

```
struct sockaddr_in {
        short   sin_family;
        u_short sin_port;
        struct  in_addr sin_addr;
        char    sin_zero[8];
};
```

IPv6 sockaddr_in6

```
struct sockaddr_in6 {
        short   sin6_family;
        u_short sin6_port;
        u_long  sin6_flowinfo;
        struct  in6_addr sin6_addr;
        u_long  sin6_scope_id;
};
```

SOCKADDR

Monitored Processes

C:\Windows\System32\svchost.exe - PID: 1068
C:\Windows\System32\rdpclip.exe - PID: 8260

Summary | 3 of 19,642 calls | 99% filtered out | 12.76 MB used | svchost.exe

#	Time of Day	Thread	Module	API	Return Value	Error	Duration
190	11:02:43.598 PM	5	RDPBASE.dll	GetNameInfoW (0x000000b47acffb40, 28, "", 1025, NULL, 0, NI_NUMERICHOST)	ERROR_SUCCESS		0.0000034
193	11:02:43.598 PM	5	RDPBASE.dll	GetNameInfoW (0x000000b47acffb40, 28, "", 1025, NULL, 0, NI_NUMERICHOST)	ERROR_SUCCESS		0.0000019
291	11:02:57.411 PM	7	termsrv.dll	GetNameInfoW (0x000000b47a3ff210, 28, "", 64, NULL, 0, NI_NUMERICHOST)	ERROR_SUCCESS		0.0000084

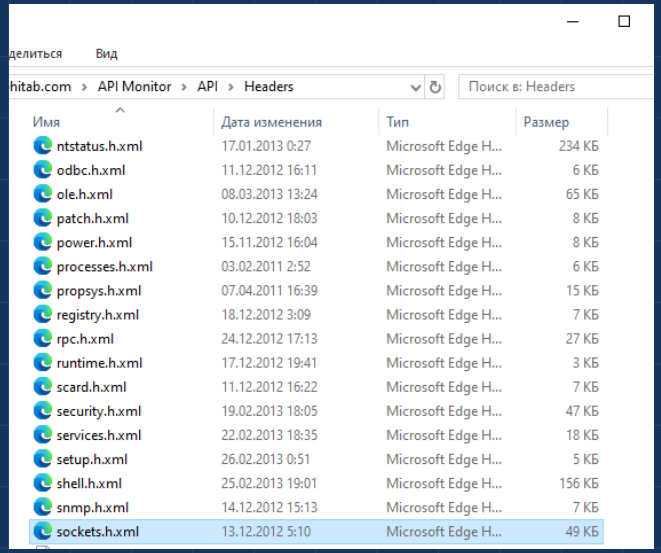
Parameters: GetNameInfoW (Ws2_32.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	const SOCKADDR*	pSockaddr	0x000000b47a3ff210	0x000000b47a3ff210
	SOCKADDR		{ sa_family = AF_INET6, sa_data = "" }	{ sa_family = AF_INET6, sa_data = "" }
	u_short	sa_family	AF_INET6	AF_INET6
	char [14]	sa_data	""	""
2	socklen_t	SockaddrLength	28	28
3	PWCHAR	pNodeBuffer	0x000001d8de7c1a20 ""	0x000001d8de7c1a20 "::%16777216"
4	DWORD	NodeBufferSize	64	64
5	PWCHAR	pServiceBuffer	NULL	NULL
6	DWORD	ServiceBufferSize	0	0
7	INT	Flags	NI_NUMERICHOST	NI_NUMERICHOST

Hex Buffer: 16 bytes (Post-Call)

0000 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

SOCKADDR



C:\Program Files\rohitab.com\API Monitor\API\Headers\sockets.h.xml

Улучшенный SOCKADDR

```
    <!-- Arrays -->
    <Variable Name="int [FD_MAX_EVENTS]"
    <Variable Name="TCHAR [WSAPROTOCOL_LEN + 1]"
    <Variable Name="DWORD [MAX_PROTOCOL_CHAIN]"
    <Variable Name="char [WSADESCRIPTION_LEN + 1]"
    <Variable Name="char [WSASYS_STATUS_LEN + 1]"
    <Variable Name="char [14]"
```

```
    <!-- Arrays -->
    <Variable Name="int [FD_MAX_EVENTS]"                Type="Array"   Base="int"     Count="10" />
    <Variable Name="TCHAR [WSAPROTOCOL_LEN + 1]"        Type="Array"   Base="TCHAR"   Count="256" />
    <Variable Name="DWORD [MAX_PROTOCOL_CHAIN]"         Type="Array"   Base="DWORD"   Count="7" />
    <Variable Name="char [WSADESCRIPTION_LEN + 1]"      Type="Array"   Base="char"    Count="257" />
    <Variable Name="char [WSASYS_STATUS_LEN + 1]"       Type="Array"   Base="char"    Count="129" />
    <Variable Name="char [14]"                          Type="Array"   Base="char"    Count="14" />
    <Variable Name="char [30]"                          Type="Array"   Base="char"    Count="30" />
```

```
    <!-- struct sockaddr -->
    <Variable Name="struct sockaddr"            Type="Struct">
        <Field Type="[ADDRESS_FAMILY_ushort]"   Name="sa_family" />
        <Field Type="char [14]"                 Name="sa_data" />
    </Variable>
    <Variable Name="struct sockaddr*"       Type="Pointer"  Base="struct socka
    <Variable Name="const sockaddr*"        Type="Pointer"  Base="struct socka
    <Variable Name="const struct sockaddr*" Type="Pointer"  Base="struct socka
```

```
    <!-- struct sockaddr -->
    <Variable Name="struct sockaddr"            Type="Struct">
        <Field Type="[ADDRESS_FAMILY_ushort]"   Name="sa_family" />
        <Field Type="char [30]"                 Name="sa_data" />
    </Variable>
    <Variable Name="struct sockaddr*"       Type="Pointer"  Base="struct sockaddr"
    <Variable Name="const sockaddr*"        Type="Pointer"  Base="struct sockaddr"
    <Variable Name="const struct sockaddr*" Type="Pointer"  Base="struct sockaddr"
```

Повтор эксперимента

Теперь наглядно видно, почему получается именно ::%16777216

17	11:06:07.243 PM	38	termsrv.dll	GetNameInfoW (0x000000acdf8ff320, 16, "", 64, NULL, 0, NI_NUMERICHOST)	ERROR_SUCCESS	0.0000045
18	11:06:07.248 PM	13	RDPBASE.dll	GetNameInfoW (0x000002186053dff0, 128, "", 1025, NULL, 0, NI_NUMERICHOST)	ERROR_SUCCESS	0.0000054
19	11:06:07.248 PM	13	RDPBASE.dll	GetNameInfoW (0x00000218605c73f0, 128, "", 1025, NULL, 0, NI_NUMERICHOST)	ERROR_SUCCESS	0.0000016

Pre-Call Value	Post-Call Value
0x000000ace25ff890	0x000000ace25ff890
{ sa_family = AF_INET6, sa_data = ...	{ sa_family = AF_INET6, sa_data = "" }
AF_INET6	AF_INET6
""	""
28	28
0x000002186a141cb0 ""	0x000002186a141cb0 "::%16777216"
64	64
NULL	NULL
0	0

Hex Buffer: 30 bytes (Pre-Call)

```
0000  17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01
```

```
struct sockaddr_in6 {
    short   sin6_family;
    u_short sin6_port;
    u_long  sin6_flowinfo;
    struct  in6_addr sin6_addr;
    u_long  sin6_scope_id;
};
```

Calculator

Programmer

16 777 216

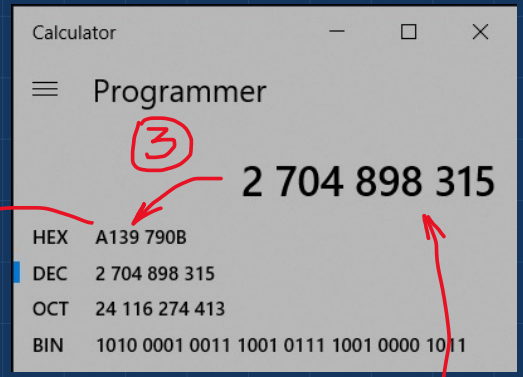
HEX 100 0000
DEC 16 777 216
OCT 100 000 000
BIN 0001 0000 0000 0000 0000 0000 0000

Уточняем гипотезу

На вход **GetNameInfoW** при использовании **IPv6** по какой-то причине поступает «испорченная» структура **sockaddr_in6**, значение IP-адреса в которой «сдвинуто» на 4 байта вправо

А как проверить? Легко

```
C:\Users\Administrator.CYBERDYNE>ipconfig

Настройка протокола IP для Windows


Адаптер Ethernet Ethernet0:

   DNS-суффикс подключения . . . . . : cyberdyne.com
   Локальный IPv6-адрес канала . . . : fe80::8c59:7475:b79:39a1%7
   IPv4-адрес. . . . . . . . . . . . : 192.168.223.37
   Маска подсети . . . . . . . . . . : 255.255.255.0
   Основной шлюз. . . . . . . . . : 192.168.223.254
```

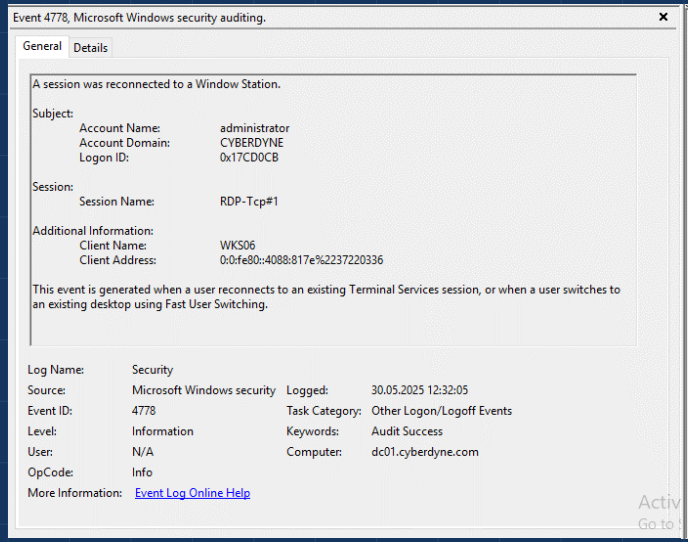


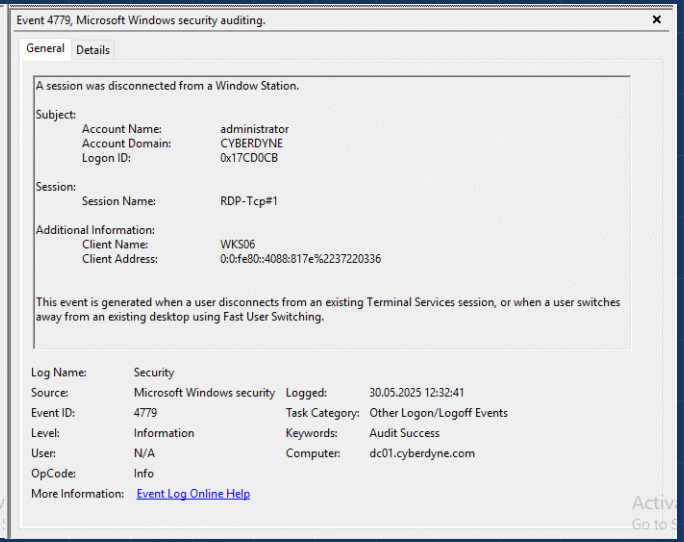

```
        },
        "UserData": {
            "EventXML": {
                "xmlns": "Event_NS",
                "Param1": "Administrator",
                "Param2": "CYBERDYNE",
                "Param3": "0:0:fe80::8c59:7475%2704898315"
            }
        }
    }
```

4 2 1

Где проявляется проблема?

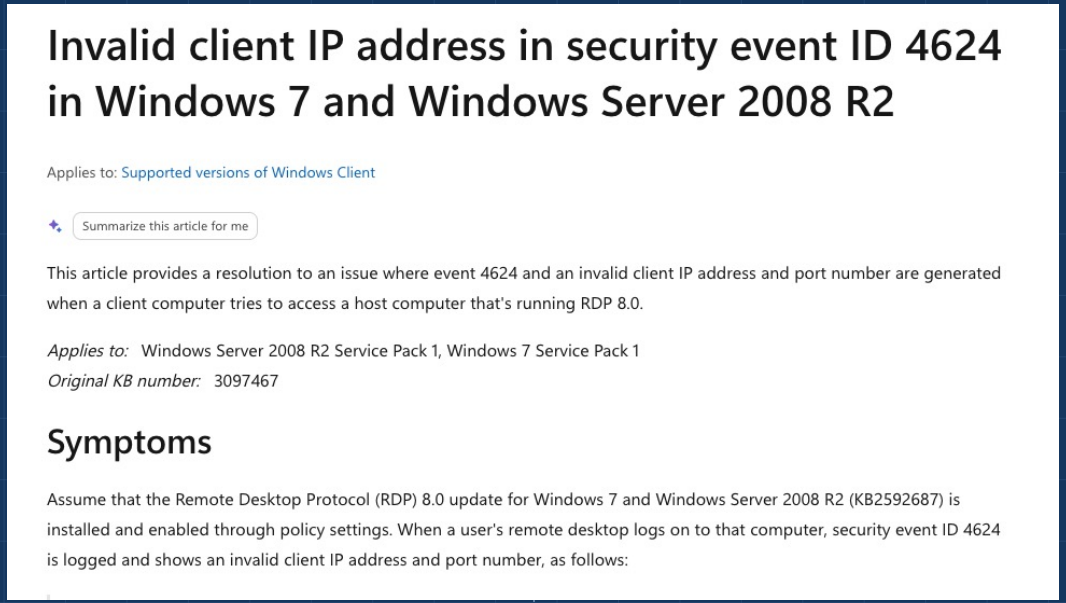
- 1149
- 4778
- 4779





Журналы приложений и служб > Microsoft > Windows > TerminalServices-RemoteConnectionManager > Operational
Windows Logs > Security

А может в интернете уже всё описано?



Invalid client IP address in security event ID 4624 in Windows 7 and Windows Server 2008 R2

Applies to: Supported versions of Windows Client

Summarize this article for me

This article provides a resolution to an issue where event 4624 and an invalid client IP address and port number are generated when a client computer tries to access a host computer that's running RDP 8.0.

Applies to: Windows Server 2008 R2 Service Pack 1, Windows 7 Service Pack 1
Original KB number: 3097467

Symptoms

Assume that the Remote Desktop Protocol (RDP) 8.0 update for Windows 7 and Windows Server 2008 R2 (KB2592687) is installed and enabled through policy settings. When a user's remote desktop logs on to that computer, security event ID 4624 is logged and shows an invalid client IP address and port number, as follows:

https://learn.microsoft.com/en-us/troubleshoot/windows-client/remote/invalid-client-ip-address-port-number-event-4624

А может в интернете уже всё описано?

Invalid client IP address in security event ID 4624

Cause

This issue occurs because of a code change in RDP 8.0. In RDP 8.0, the client IP address is stored in a WTS_SOCKADDR structure. This differs from RDP 7.0 (the default RDP version in Windows 7 and Windows Server 2008 R2).

In Windows 8 and Windows Server 2012 (and later versions of Windows), the code logic for logging this event is rewritten based on the new design. That prevents this issue from occurring.

https://learn.microsoft.com/en-us/troubleshoot/windows-client/remote/invalid-client-ip-address-port-number-event-4624

Настало время немного подебажить

Неочевидные сложности

ssh.exe - удобно, но не всегда

Арендовать виртуалку с актуальной версией Windows, чтобы быстро проверить гипотезу – пара минут

Возиться с открытием SSH для доступа снаружи – так себе идея

Арендовать две – в два раза дороже :/

Небольшие лайфхаки

У MS есть удобная утилита - devtunnels

Позволяет «пробросить» порт через Интернет, так же, как и ngrok

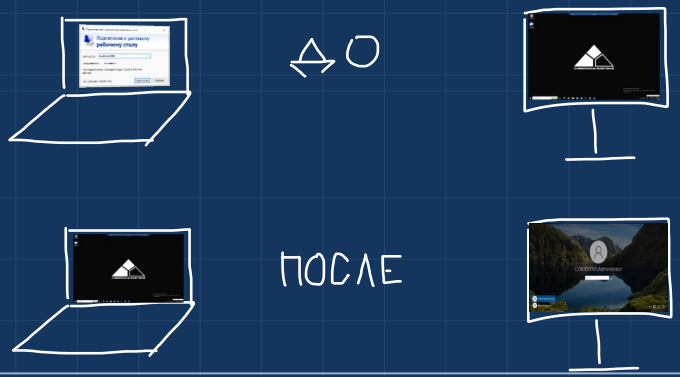
```
PS C:\Users\administrator.CYBERDYNE> .\devtunnel.exe host -p 3389
Hosting port: 3389
Connect via browser: https://x55dtvz1-3389.euw.devtunnels.ms
Inspect network activity: https://x55dtvz1-3389-inspect.euw.devtunnels.ms

Ready to accept connections for tunnel: jolly-shoe-1np4kwf.euw
```

https://learn.microsoft.com/ru-ru/azure/developer/dev-tunnels/get-started

Настало время немного подебажить

Неочевидные сложности

Точки останова в ходе удаленного подключения = «застрять в текстурах»



Небольшие лайфхаки

- многие отладчики позволяют отлаживать удаленно
- devtunnel может «пробросить» сразу два порта

```
PS C:\Users\administrator.CYBERDYNE> .\devtunnel.exe host -p 3389 -p 5000
Hosting port: 3389
Connect via browser: https://ngx914s9-3389.euw.devtunnels.ms
Inspect network activity: https://ngx914s9-3389-inspect.euw.devtunnels.ms
Hosting port: 5000
Connect via browser: https://ngx914s9.euw.devtunnels.ms:5000, https://ngx914s9-5000.euw.devtunnels.ms
Inspect network activity: https://ngx914s9-5000-inspect.euw.devtunnels.ms

Ready to accept connections for tunnel: new-horse-h5rht5p.euw
```

Настало время немного подебажить

Неочевидные сложности

- нормальный код (нет обфускации)
- есть символы (+100500 к удаче)
- кода много (-100500 к морали)

Небольшие лайфхаки

После API Monitor уже известно место, где проблема проявляется, нужно всего лишь найти чуть раньше место, где она рождается

При отладке точки останова можно ставить не только на определенное место в коде, но и на доступ к определенным участкам в памяти (hardware breakpoints)

Настало время немного подебажить

Библиотека: rdpcorets.dll
Метод: CUMRDPConnection::GetClientData
Кусок кода: F3 41 0F 7F 86 10 0C 00 movdqu xmmword ptr [r14+3088], xmm0

Заносит 128 бит IPv6-адреса клиента из регистра xmm0 в память по смещению **r14+3088**
Выше виден код, который заносит значение 17h по смещению **r14+3076**
3088 - 3076 = 12, запомним это

```
                        loc_1801C73D7:                          ; CODE XREF: CUMRDPConnection::GetClientData(_WTS_CLIENT_DATA *)+A298C↑j
0F 10 45 E8                           movups  xmm0, xmmword ptr [rbp+var_18.u]
B8 17 00 00 00                        mov     eax, 17h
66 41 89 86 04 0C 00 00               mov     [r14+3076], ax
F3 41 0F 7F 86 10 0C 00               movdqu  xmmword ptr [r14+3088], xmm0
00
E9 5D D7 F5 FF                        jmp     loc_180124B53
```

Та самая структура WTS_SOCKADDR

Нетрудно заметить

- 17h соответствует константе AF_INET6
- в этом месте видим заполнение WTS_SOCKADDR данными об IPv6-адресе клиента
- в структуре WTS_SOCKADDR 4 «лишних» байта выделены для выравнивания

```
00000000 struct _WTS_SOCKADDR // sizeof=0x20
00000000 {
00000000     USHORT sin_family;
00000002     // padding byte
00000003     // padding byte
00000004     union _WTS_SOCKADDR::$2288C3058969AF37A9D9C0F15090F76C u;
00000020 };

00000000 union _WTS_SOCKADDR::$2288C3058969AF37A9D9C0F15090F76C // sizeof=
00000000 {
00000000     struct _WTS_SOCKADDR::$2288C3058969AF37A9D9C0F15090F76C::$F2E
00000000     struct _WTS_SOCKADDR::$2288C3058969AF37A9D9C0F15090F76C::$235
00000000 };

00000000 struct _WTS_SOCKADDR::$2288C3058969AF37A9D9C0F15090F76C::$2353073
00000000 {
00000000     USHORT sin6_port;
00000002     // padding byte
00000003     // padding byte
00000004     ULONG sin6_flowinfo;
00000008     USHORT sin6_addr[8];
00000018     ULONG sin6_scope_id;
0000001C };
```

Парсинг данных как sockaddr_in6

строка для записи в лог формируется в коде библиотеки termsrv.dll при помощи вызова метода GetNameInfoW из библиотеки ws2_32.dll

в качестве параметра ожидается указатель на SOCKADDR

в случае IPv6 (sin_family = 17h) используется смещение 8 байт – и это на 4 меньше, чем 12

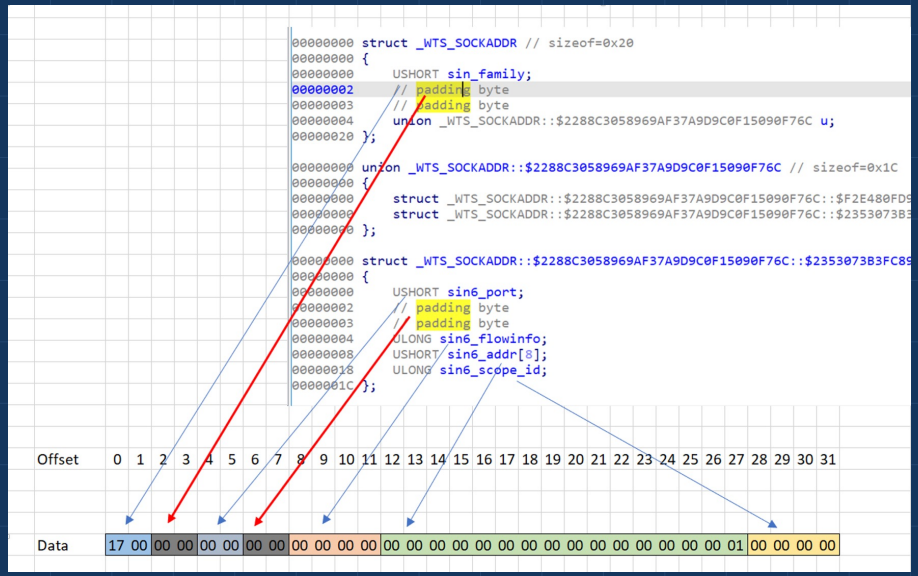
```
                                loc_18000FA83:
48 85 FF                        test    rdi, rdi
0F 84 01 7F 00 00               jz      loc_18001798D
0F B7 07                        movzx   eax, word ptr [r
83 F8 17                        cmp     eax, 17h
0F 85 A1 00 00 00               jnz     loc_18000FB39
8D 68 05                        lea     ebp, [rax+5]
4C 8D 4F 08                     lea     r9, [rdi+8]
8D 48 F9                        lea     ecx, [rax-7]
```

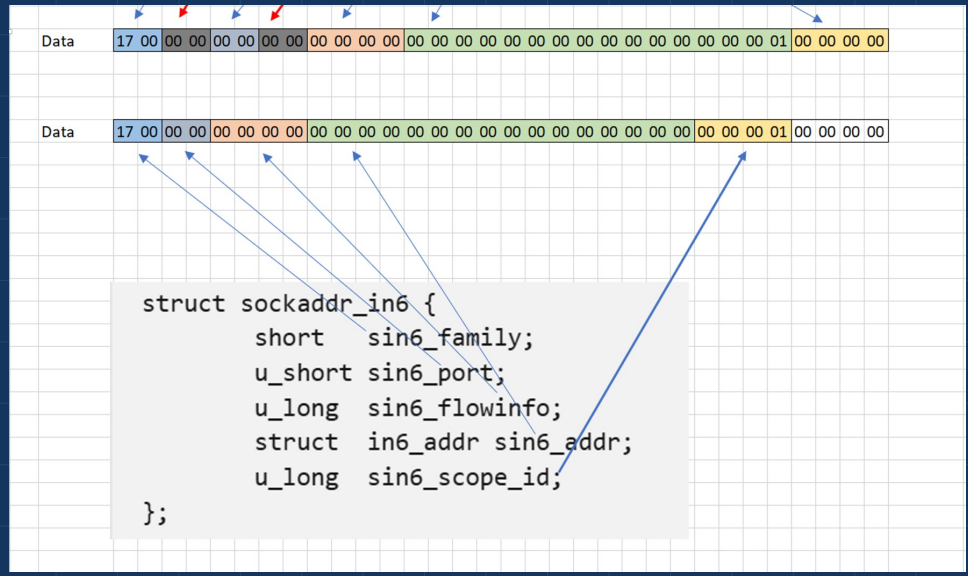
C++

```
typedef struct sockaddr_in6 {
  ADDRESS_FAMILY sin6_family;
  USHORT         sin6_port;
  ULONG          sin6_flowinfo;
  IN6_ADDR       sin6_addr;
  union {
    ULONG    sin6_scope_id;
    SCOPE_ID sin6_scope_struct;
  };
} SOCKADDR_IN6_LH, *PSOCKADDR_IN6_LH, *LPSOCKADDR_IN6_LH;
```

2 байта
2 байта
4 байта

Картинка, чтобы стало понятнее





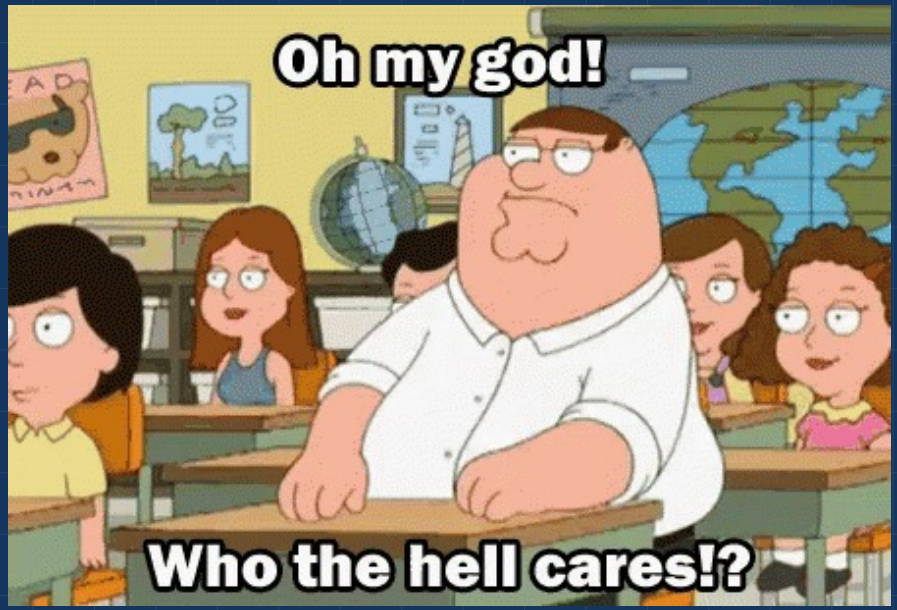
rdpcorets.dll

termsrv.dll, ws2_32.dll

Какие версии содержат ошибку

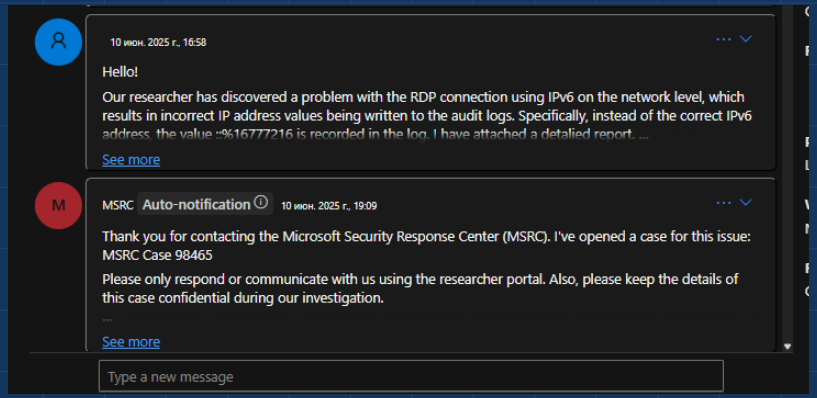
- Windows Server 2012 R2
- Windows 10 Pro 22H2 19045.5854
- Windows Server 2019 Standard 1809
- Windows 11 Enterprise 23H2 22631.5335
- Скорее всего все версии,
  начиная с Windows 8
  и Windows Server 2012

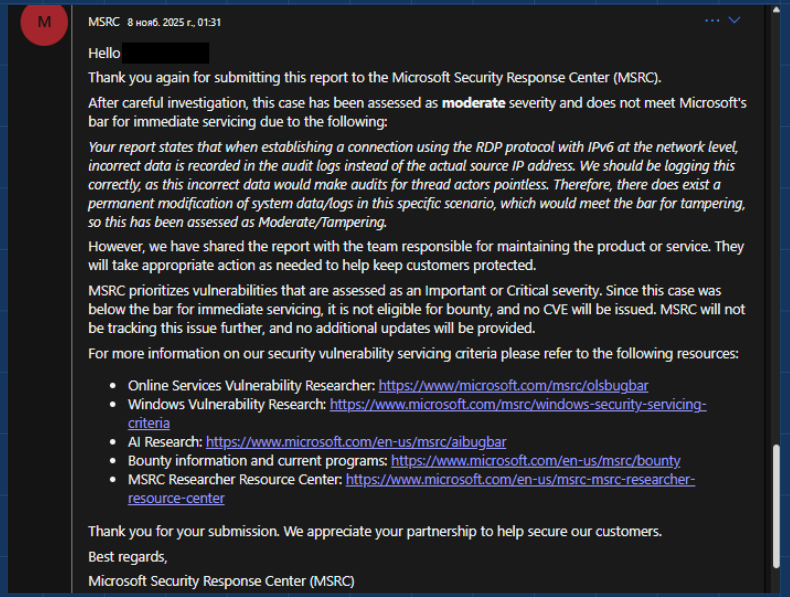
данные актуальны на 10.06.2025



Хронология исправления

- 10.06.2025 – репорт в MSRC
- 08.11.2025 – ответ: moderate, no CVE, no additional updates will be provided





Не может быть, чтобы никто не жаловался

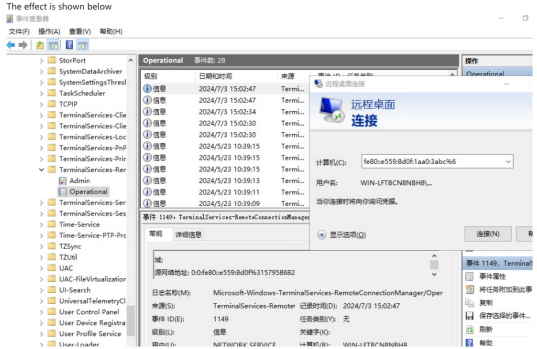
12 Jul 2024, 09:04

Is there translation logic for IPV6 addresses in external access logs in windows?

Anonymous 12 Jul 2024, 09:04

In the case of ipv6, how does the login source address format with event ID 1149 in Microsoft-Windows-TerminalServices-RemoteConnectionManager be converted to the address queried by the client using the ipconfig command. For example, the address I entered on the client is [2001::311], but the Microsoft-Windows-TerminalServices-RemoteConnectionManager records 0:0:2001::%285409280.

The effect is shown below



Windows for business | Windows Client for IT Pros | User experience | Other

29 Apr 2026, 15:21

2 answers

Andreas Mai 29 Apr 2026, 15:21
0

This seems to be a bug within Windows.

The IPv6 address gets shifted to the right by 32 bit. The overflow gets put in the "Scope ID".

In your case: 2001::311 = **2001:0000:0000:0000:0000:0000**:0000:0311

0:0:2001::%285409280 = 0000:0000:**2001:0000:0000:0000:0000:0000** (%285409280)
shift 32 bit to left => **2001:0000:0000:0000:0000:0000**:xxxx:xxxx

285409280 (DEC) = 11030000 (HEX) => in little endian 0000:0311 which equals the rightmost 32 bit

I'm currently writing a bugreport for this.

Was this answer helpful? Yes No

0 comments Report a concern

Sign in to comment

https://learn.microsoft.com/en-ie/answers/questions/1810270/is-there-translation-logic-for-ipv6-addresses-in-e

А может уже поправили? Поправили

Есть ошибка (на 10.06.2025):

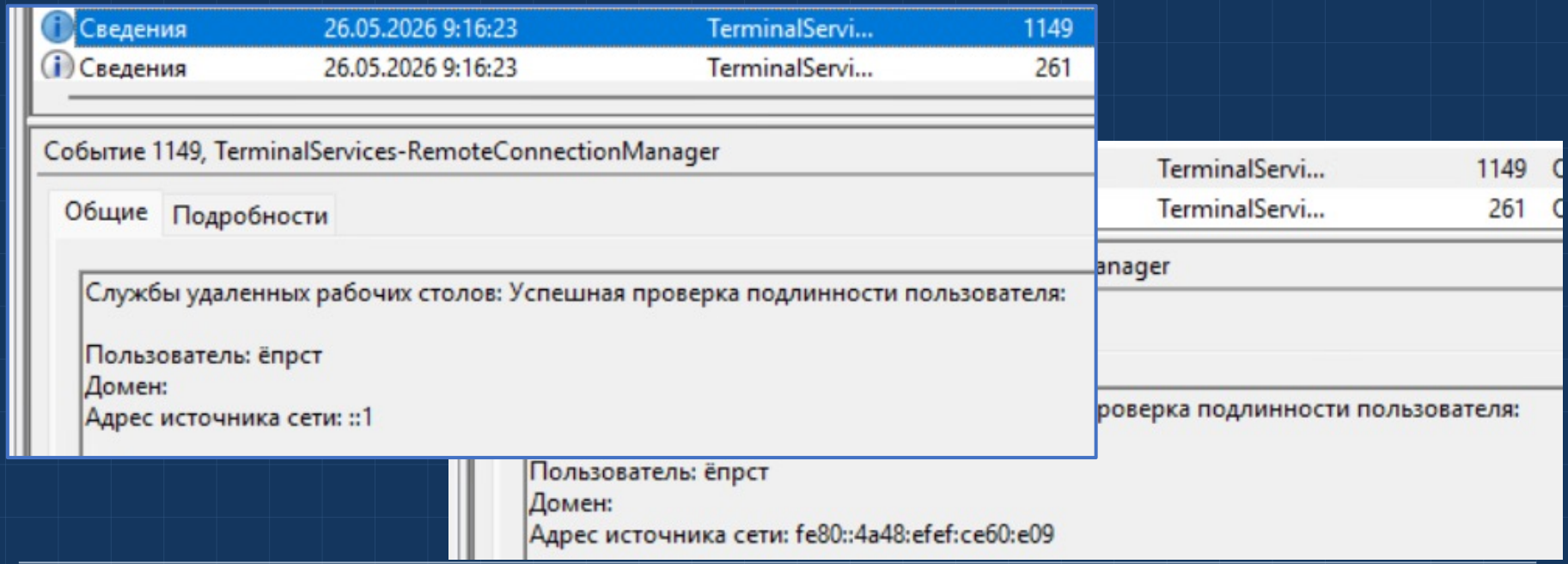
- Windows Server 2012 R2
- Windows 10 Pro 22H2 19045.5854
- Windows Server 2019 Standard 1809
- Windows 11 Enterprise 23H2 22631.5335
- Скорее всего все версии, начиная с Windows 8 и Windows Server 2012*

На 25.05.2026:

- Windows 10 Pro 22H2 19045.6456 – есть ошибка
- Windows 11 24H2 26100.1742 – есть ошибка

- Windows 11 24H2 26100.8457 – нет ошибки
- Windows 11 26H1 28000.2113 – нет ошибки

А может уже поправили? Поправили



Итоги

::%16777216 – результат ошибки Windows

ошибка в несогласованной обработке одних и тех же данных разными модулями

из-за ошибки при любых подключениях к RDP с использованием IPv6 в лог попадет неверный адрес источника

ошибка не мешает использовать ::%16777216 как IoC для выявления подключений к RDP по IPv6 через туннель (любых, не только ngrok)

ошибочное значение из лога можно восстановить и использовать при сборе событий в SIEM, в корреляциях, при тредхантинге и т.д.

БОНУС

Как восстановить нормальный IPv6-адрес

1. Взять значение из лога:

`0:0:fe80::6e1d:980d%2607874452`

2. Часть после % перевести в HEX

`2 607 874 452 dec = 9b 71 01 94 hex`

3. Убрать слева два нуля, дописать справа полученные цифры с учетом обратного порядка байт

`fe80::6e1d:980d:9401:719b`

Под силу реализовать практически в любой SIEM или иной системе для обработки журналов аудита

– SLIDE 41 / END –

QUESTIONS · COMMENTS · DISAGREEMENTS

Константин Грищенко ·Positive Technologies