

CHAITIN

Remote exploitation of the Valve Source game engine

Amat Cama

Agenda

- Introduction
- Prior Work
- Motivation
- Game Engine ?
- Valve Source Engine
- Hunting for Bugs
- Conclusion and Future Work



Introduction



Introduction

Amat Cama

- Senior Security Researcher at Chaitin Tech
- CTF player, team Shellphish
- Pwning
- asdf

Beijing Chaitin Tech Co., Ltd:

- chaitin.cn/en
- Pentesting services and enterprise products
- **D-Sensor** - Threat Perception System
- **SafeLine** - Web Application Firewall
- Chaitin Security Research Lab:
 - GeekPwn 2016 awardees: PS4 Jailbreak, Android rooting, Router rooting, <this project>
 - Pwn20wn 2017 3rd place: Safari + root, Firefox + SYSTEM, Ubuntu root, macOS root
 - CTF players from team blo0p, 2nd place at DEFCON 2016



Prior Work



Prior Work

- **“Multiplayer Online Games Insecurity”** - *Luigi Ariemma & Donato Ferrante*:
 - <https://media.blackhat.com/eu-13/briefings/Ferrante/bh-eu-13-multiplayer-online-games-ferrante-wp.pdf>
 - Ideas and methodologies for attacking game engines
- **“Exploiting Game Engines For Fun & Profit”** - *Luigi Ariemma & Donato Ferrante*:
 - https://revuln.com/files/Ferrante_Ariemma_Exploiting_Game_Engines.pdf
 - Study of a number of games and game engines
 - Number of bugs
- **“Game Engines: A 0-day’s tale”** - *Luigi Ariemma & Donato Ferrante*
 - http://revuln.com/files/ReVuln_Game_Engines_0days_tale.pdf
 - Describes a number of bugs in different games and game engines
- ...



Motivation



Motivation

- Because it is fun
- Fairly new area of research; great opportunity to learn
- Millions of players every day
- How hard would it be to get hacked by just connecting to a game server ?



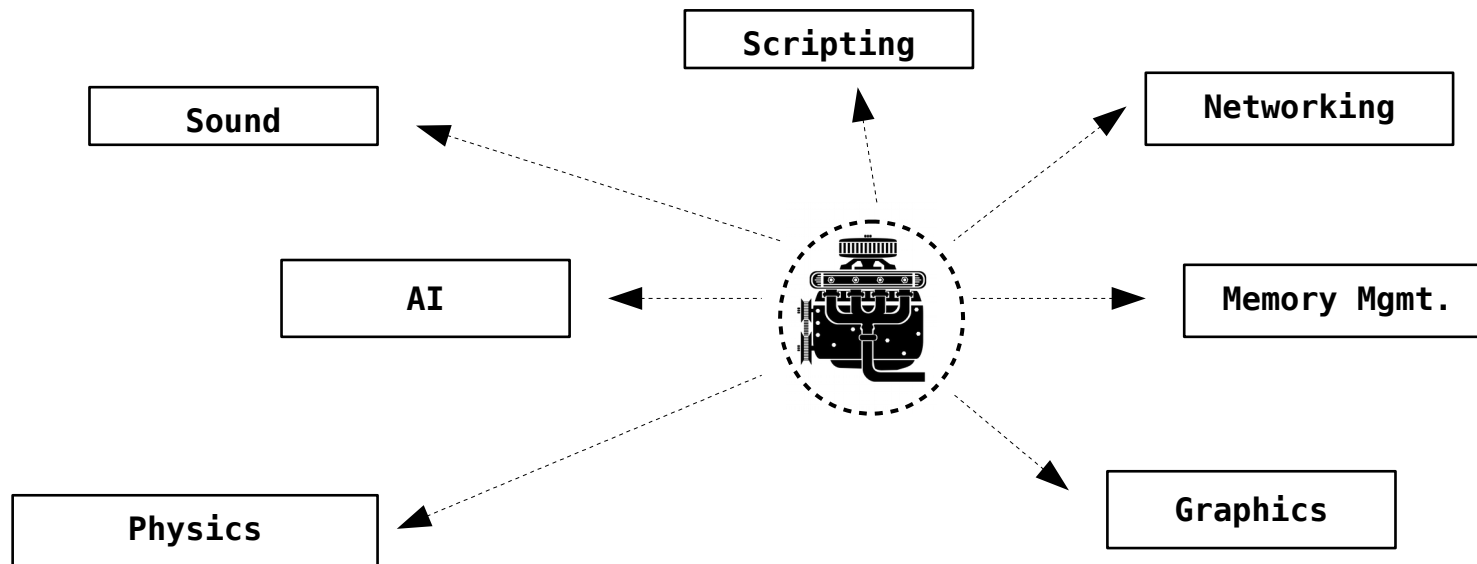
Game Engine ?



Game Engine ?

What is a Game Engine (I) ?

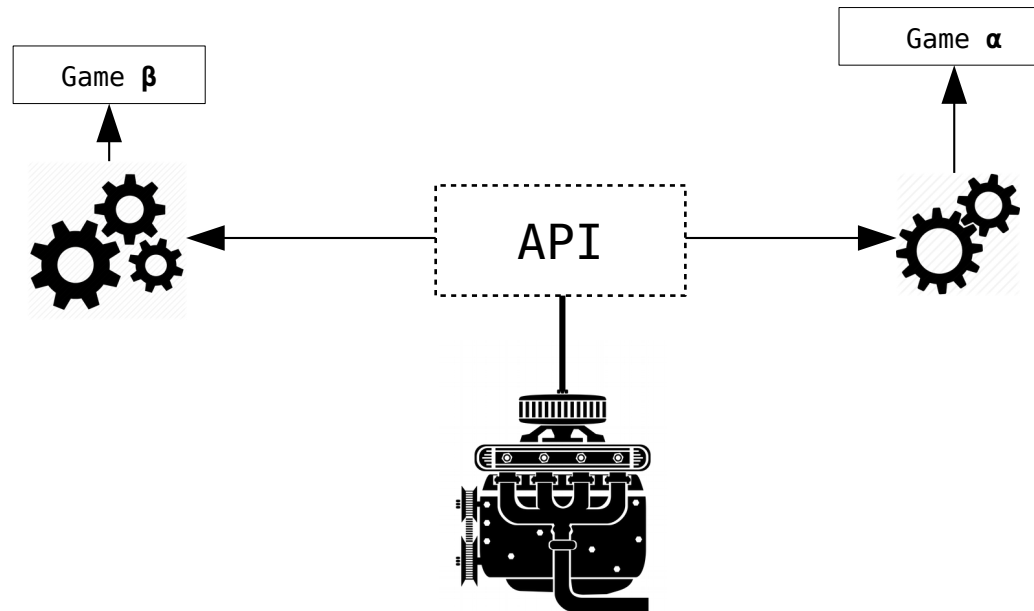
- TL;DR: The “Kernel” of the game
- Software framework designed for the creation and development of video games
- Reusable code that provides functionality such as a renderer for 2D or 3D graphics, physics, sound, scripting, networking...



Game Engine ?

What is a Game Engine (II) ?

- Provides APIs to perform different operations
- SDKs built on top of these APIs
- Highly customizable to allow making different game types (FPS, RTS, etc.) using the same engine



Game Engine ?

Popular Game Engines

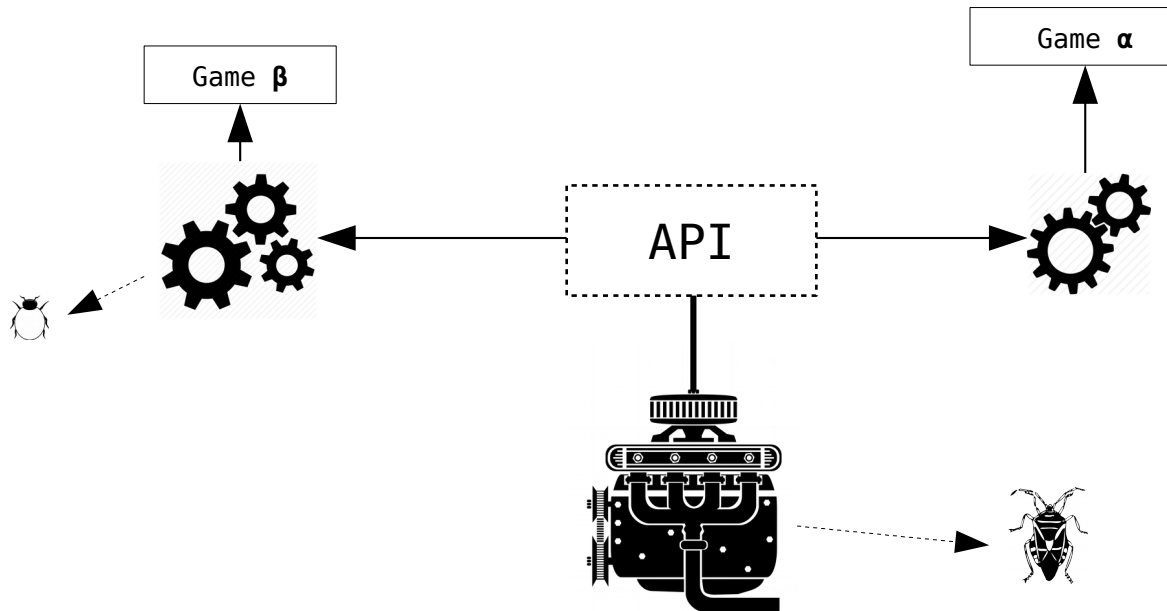
- Frostbite: Battlefield, Army of Two, Fifa
- Unreal Engine: Unreal Tournament, X-COM, Bioshock
- **Source:** Team Fortress, Counter Strike, Dota
- Unity: Temple Run, Pwn Adventure 3, <other rumored ctf challenge> ^^
- CryEngine: Far Cry, Enemy Front, Crysis
- ...



Game Engine ?

Bugs ?

- For performance reasons, a number of game engines written in C/C++
- Bugs in custom game code are **interesting**
- Bugs in engine core are **more interesting**



Valve Source Engine



Valve Source Engine

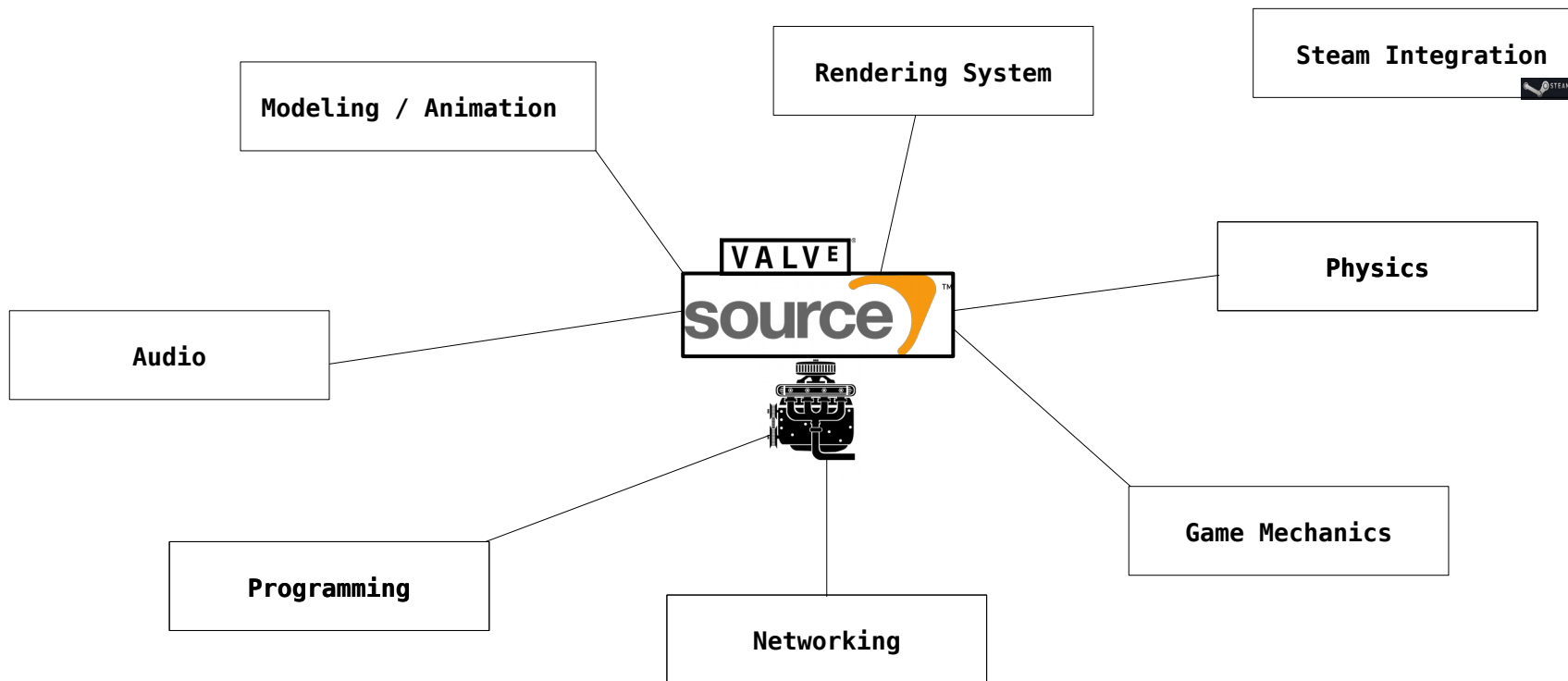
About the engine

- Relatively old; has its roots in original Half-Life released in 1998
- Popular game engine



Valve Source Engine

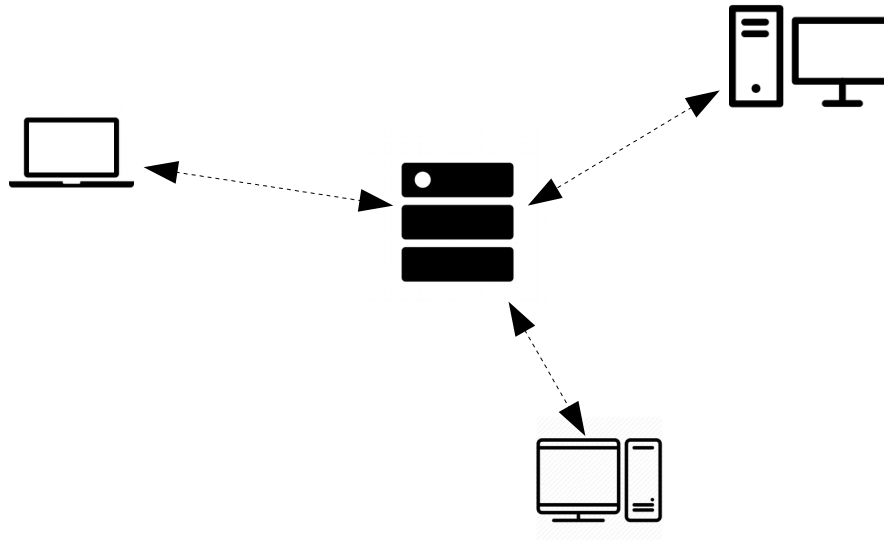
Architecture and Features



Valve Source Engine

Multiplayer Networking (I)

- Client-Server networking architecture
- Server in charge of world simulation, game rules, and player input processing
- Client connects to server and “obeys” orders
- Communication is done through UDP/IP



Multiplayer Networking (II)

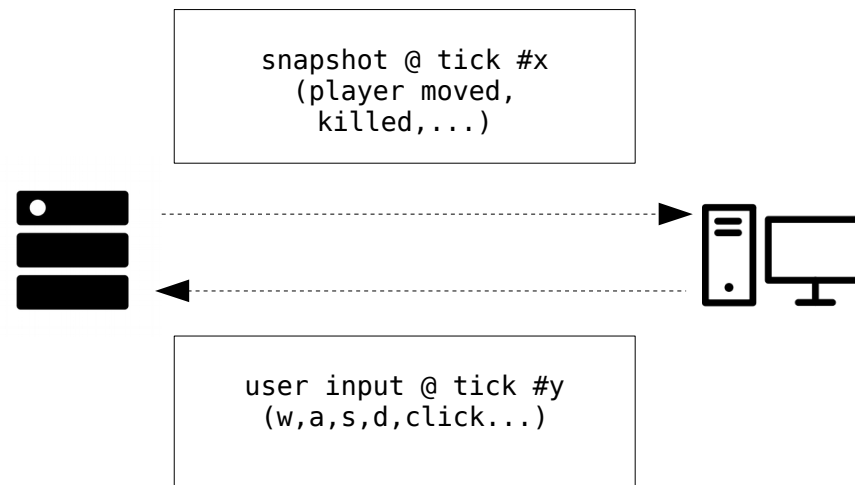
- UDP is preferred due to speed
- Reimpliment of “TCP” over UDP
- Packet Splitting, Fragmentation, Reassembly...
- Compression and Decompression
- Encryption
- Somewhat complex processing, might low hanging fruits in these handlers



Valve Source Engine

Multiplayer Networking (III)

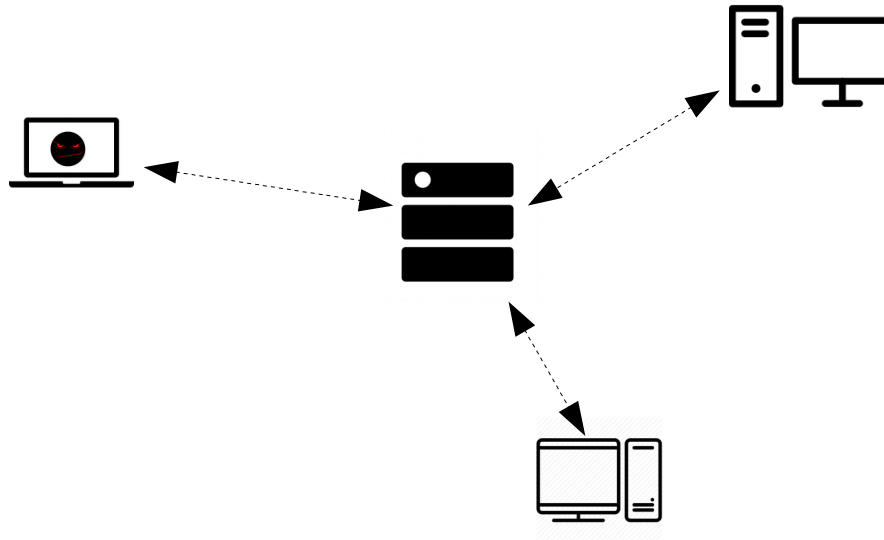
- Game is simulated at fixed time intervals: *ticks*
- At each tick, server takes a *snapshot* of the world and sends it to the clients
- Snapshots contain information about game elements and world that have changed since the previous tick
- Clients sample user input (mouse, keyboard) and send to server



Valve Source Engine

Messages (I)

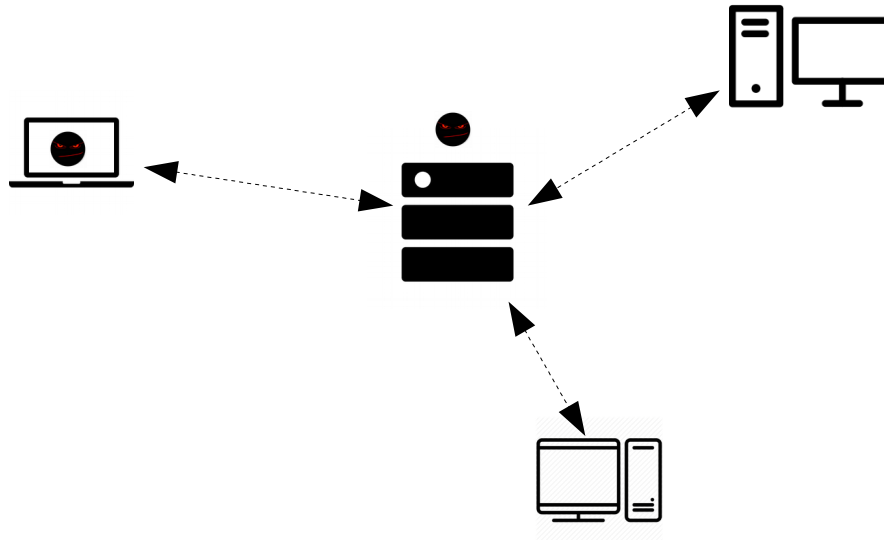
- Mainly 3 categories of messages sent between client and server: *Bidirectional*, *Client* and *Server* messages.
- If looking for RCE type bugs, interesting place to look in
- Can find bugs in Client



Valve Source Engine

Messages (I)

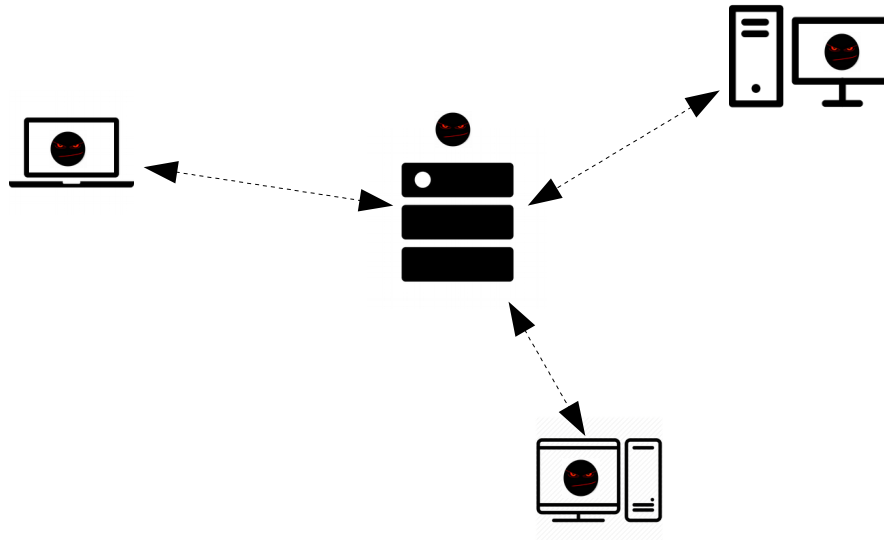
- Mainly 3 categories of messages sent between client and server: *Bidirectional*, *Client* and *Server* messages.
- If looking for RCE type bugs, interesting place to look in
- Can find bugs in Client
- Can find bugs in Server



Valve Source Engine

Messages (I)

- Mainly 3 categories of messages sent between client and server: *Bidirectional*, *Client* and *Server* messages.
- If looking for RCE type bugs, interesting place to look in
- Can find bugs in Client
- Can find bugs in Server
- Can find bugs that affect both Client and Server



Valve Source Engine

Bidirectional Messages

- Sent by both Client and Server, “net_X”:
 - net_NOP
 - net_Disconnect
 - net_File
 - net_LastControlMessage
 - net_SplitScreenUser
 - **net_Tick** ⓘ
 - **net_StringCmd** ⓘ
 - net_SetConVar
 - net_SignonState



Valve Source Engine

Client Messages

- Sent by the Client, "clc_X":
 - clc_ClientInfo
 - clc_Move
 - clc_VoiceData
 - clc_BaselineAck
 - clc_ListenEvents
 - **clc_RespondCvarValue** ⓘ
 - clc_FileCRCCheck
 - clc_LoadingProgress
 - clc_SplitPlayerConnect
 - clc_ClientMessage
 - clc_CmdKeyValues



Valve Source Engine

Server Messages (I)

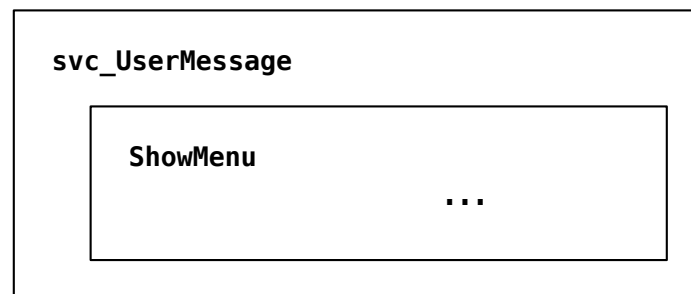
- Sent by the server, “svc_X”:
 - svc_ServerInfo
 - svc_SendTable
 - **svc_CreateStringTable** ⓘ
 - svc_UpdateStringTable
 - svc_Print
 - **svc_UserMessage** ⓘ
 - svc_EntityMessage
 - svc_GameEvent
 - **svc_PacketEntities** ⓘ
 - svc_TempEntities
 - svc_Prefetch
 - svc_GameEventList
 - **svc_GetCvarValue** ⓘ
 - svc_CmdKeyValues
 - ...



Valve Source Engine

Server Messages (II)

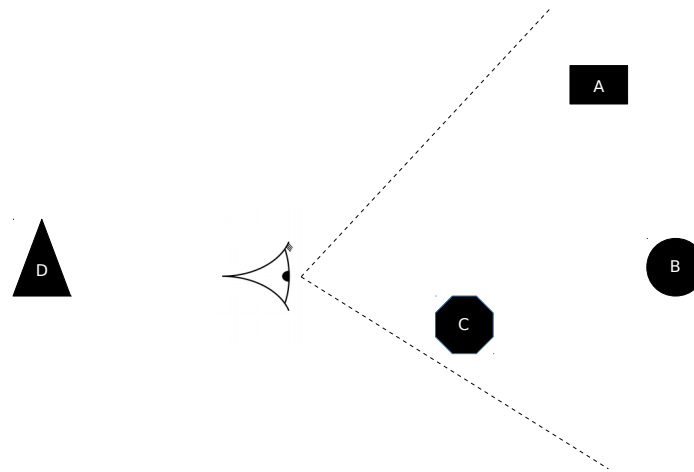
- **svc_UserMessage**'s are game specific messages that are used to notify the client of different game events
- Sub-message type of Server Messages
- Some are shared between different games
- Vulnerabilities here have a higher chance of not being present in all games
- E.g:
 - **CS_UM_ShowMenu** / **TF_UM_ShowMenu** ⓘ
 - **CS_UM_ProcessSpottedEntityUpdate** ⓘ
 - ...



Valve Source Engine

Network Entities (I)

- **svc_PacketEntities** are messages related to *Network Entities*
- Logical and physical objects (i.e almost everything) in the game world
- Entity networking system makes sure that these objects are sync'ed for all players
- Only entities that are of possible interest for a client (visible, audible etc.) are updated



Network Entities (II)

- Entities exist in both the server and clients
- When the server sends an `svc_PacketEntities` message, entities that don't exist on the client are automatically created
- Source servers can handle a max of 2048 entities
- Sent in the snapshots



Valve Source Engine

ConVars and ConCommands (I)

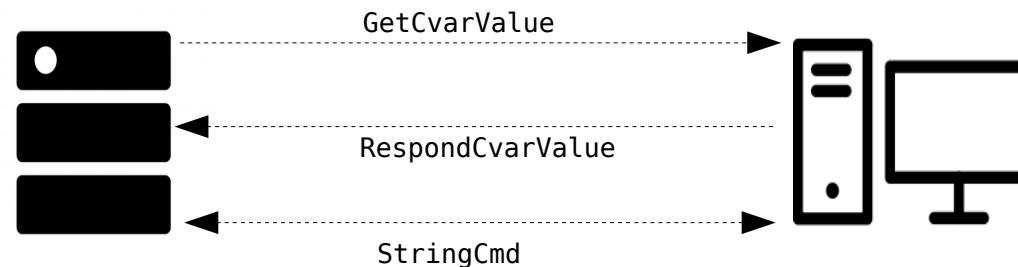
- *Console Variables* (CVars / ConVars) hold configuration parameters and are used on the server and the clients
- Some CVars are synchronized between client and server
- Some can be set by the server, others by the client
- e.g:
 - bot_dont_shoot: if set, bots won't fire their weapons
 - host_map: current map name
- *Console Commands* (ConCommands) are commands used to perform some tasks related to the game or for debugging purposes
- They are actually CVars of type 'cmd'
- e.g:
 - retry: retry connecting to the server
 - gods: all players become invulnerable
- All available ConVars and ConCommands can be viewed by issuing the cvarlist command on the console



Valve Source Engine

ConVars and ConCommands (II)

- `svc_GetCvarValue` messages can be sent by the server to request the value of a CVar
- `clc_RespondCvarValue` are the replies sent by the client
- `net_StringCmd` can be sent by both the client and server to run some ConCommands



Valve Source Engine

Message encoding, transmission, reception

- By default Source packets are encoded in bitstreams
- However games can customize how packets are encoded
- e.g: CS:GO uses Google Protocol Buffers instead of the default bitstream mechanism
- Things such as encryption and compression are optional



Hunting for Bugs in Source



Hunting for Bugs

Where to look ?

- As seen in previous slides, many places to look
- Messages are very interesting
- engine.dll - Engine core, “net_X” and “svc_X” message handlers
- client.dll – A lot of game specific code, “svc_UserMessage” handlers
- Other areas exist:
 - Game map parser; custom file format called BSP, maps are automatically downloaded from server and parsed by the client
 - MOTD (Message of the day); displayed after connecting to the server, can be a web page
 - Audio parser
 - ...



Hunting for Bugs



Int Overflow in ProcessCreateStringTable() TF2

- The `svc_CreateStringTable` server message is used to create string tables
- String tables are used to avoid sending common strings or binary blobs over and over
- Only an index is sent to address the data



Hunting for Bugs




Int Overflow in ProcessCreateStringTable() TF2

```
char CBaseClient::ProcessCreateStringTable(void *this, SVC_CreateStringTable *creatstrtab)
{
    ...
    int dataBits; // edx@2
    void *dest; // edi@8
    char *source; // esi@8
    ...
    int destLen; // [esp+30h] [ebp-Ch]@8
    int sourceLen; // [esp+38h] [ebp-4h]@6
    ...
    if ( creatstrtab->isCompressed )
    {
        dataIn = &creatstrtab->m_DataIn;
        ...
        destLen = READ32(dataIn);
        sourceLen = READ32(dataIn);
        ...
        dest = operator new[]((destLen + 3) & 0xFFFFFFFFFC); // # 1
        source = operator new[]((sourceLen + 3) & 0xFFFFFFFFFC); // # 2
        bf_read::ReadBits(dataIn, source, 8 * sourceLen);
        NET_BufferToBufferDecompress(dest, &destLen, source, sourceLen);
        ...
        operator delete[](dest);
        operator delete[](source);
    }
    ...
}
```



Hunting for Bugs

Int Overflow in ProcessCreateStringTable() TF2

- Can we exploit ?
- Corrupt adjacent heap chunks
- Memory allocator is tcmalloc, it is possible to make it return arbitrary addresses (c.f “Exploit Necromancy in TCMalloc”)
- But sadly bug not present in CS:GO 
- Let's keep looking...



Hunting for Bugs

 00B Write in UM_ProcessSpottedEntityUpdate() UserMessage handler in CS:GO

- The CS_UM_ProcessSpottedEntityUpdate user message is used to send information about some “entities” in CS:GO (tbh we don’t really care what it does)



Hunting for Bugs



00B Write in UM_ProcessSpottedEntityUpdate() UserMessage handler in CS:GO

```
int ProcessSpottedEntityUpdate(_BYTE *this, ProcessSpottedEntityUpdate_t *data)
{
    ...
    for ( i = 0; idx < data->numEntities; i = idx )
    {
        entitiesArray = data->entitiesArray;
        entName = 0;
        update = entitiesArray[idx];
        ent_idx = update->ent_idx;
        ...
        {
            ...
            objidx = 0x1E0 * ent_idx;
            *&this_[objidx - 16] = 4 * update->origin_z;
            *&this_[objidx - 24] = 4 * update->origin_x;
            *&this_[objidx - 20] = 4 * update->origin_z;
            *&this_[objidx - 12] = 0;
            *&this_[objidx - 8] = update->angle_y;
            *&this_[objidx - 4] = 0;
        }
        ...
    }
    ...
}
...
```



Hunting for Bugs

 00B Write in `UM_ProcessSpottedEntityUpdate()` UserMessage handler in CS:GO

- Can we exploit ?
- Overwrite adjacent objects, again tcmalloc header corruption
- Corrupt C++ objects on the heap
- But sadly bug not present in TF2
- Let's keep looking...



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- Network Entities described earlier
- Server sends an `svc_PacketEntities` message to the client with a snapshot of the different entities that need to be updated or created



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

```
int CL_CopyNewEntity(CEntityReadInfo *u, int iClass, int iSerialNum)
{
    ...
    signed int newEntity; // edx@1
    ...
    IClientNetworkable *ent; // edi@3
    ...

    newEntity = u->m_nNewEntity;
    if ( newEntity >= 2048 )
        return Host_Error("CL_CopyNewEntity: m_nNewEntity >= MAX_EDICTS");
    ent = entitylist->vtbl->GetClientNetworkable(newEntity);
    if ( iClass >= gClassMaxIndex )
        return Host_Error("CL_CopyNewEntity: invalid class index (%d).\n", iClass);
    ...
    if ( ent )
    {
        v8 = ent->vtbl->someMethod(ent);
        ...
    }
    ...
}
```



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

```
int CL_CopyNewEntity(CEntityReadInfo *u, int iClass, int iSerialNum)
{
    ...
    signed int newEntity; // edx@1
    ...
    IClientNetworkable *ent; // edi@3
    ...


    new IClientNetworkable * GetClientNetworkable(CClientEntityList *this, int index)
    if {
        return (&this->m_EntityCacheInfo)[2 * index];
    }
    ent }
    if ( iClass >= gclassmaxindex )
        return Host_Error("CL_CopyNewEntity: invalid class index (%d).\n", iClass);
    ...
    if ( ent )
    {
        v8 = ent->vtbl->someMethod(ent);
        ...
    }
    ...
}
```



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- Can we exploit ?
- entitylist is a global variable in client.dll and its m_EntityCacheInfo field is a static buffer of size 2048
- Can retrieve a C++ object through 00B indexing and call a method from it
- Moreover bug is present in engine core ! 



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- We want entitylist.m_EntityCacheInfo[idx] → Our fake C++ object
- Assembly code for `GetClientNetworkable` looks like:

```
GetClientNetworkable proc near
index                = dword ptr 8
    push    ebp
    mov     ebp, esp
    mov     eax, [ebp+index]
    mov     eax, [ecx+eax*8+28h]
    pop     ebp
    retn    4
GetClientNetworkable endp
```

- This means $(\text{client.dll} + \text{offset_to_array}) + \text{idx} * 8 \rightarrow \text{fake object}$
- `idx` is an integer and will be negative when `GetClientNetworkable` is called
- Since it is multiplied by 8, can make it wrap around to be a positive value
- Now we need to store controlled data in the global data section of `client.dll`

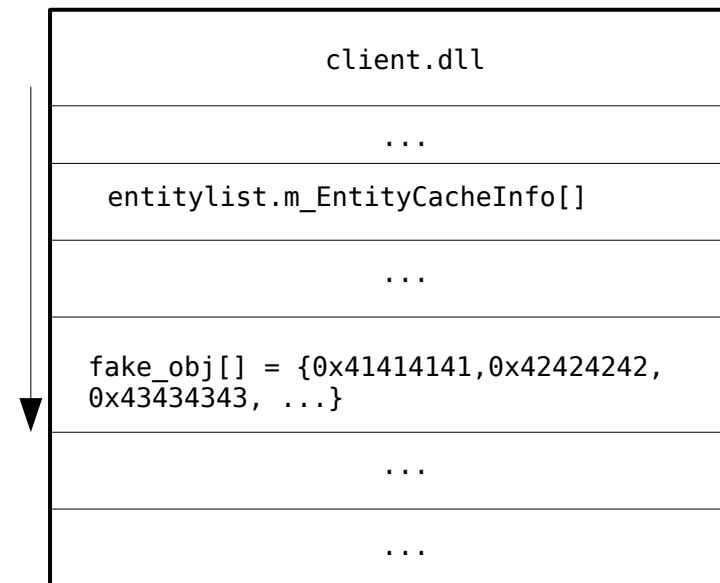


Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- How to store data ?
- Let's have a look at UserMessages
- **UM_ShowMenu** message is used to show a Hud menu on the client
- Buffers string data in a global buffer in client.dll
- So send **UM_ShowMenu** message to store fake C++ object, then trigger bug through **svc_PacketEntities**



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- “Wait, but we need to leak...”
- Let’s have a look at the rest of the `CL_CopyNewEntity()` function

```
int __cdecl CL_CopyNewEntity(CEntityReadInfo *u, int iClass, int iSerialNum)
{
    ...
    ent = CL_CreateDLLEntity(u->m_nNewEntity, iClass, iSerialNum);
    if ( !ent )
    {
        ...
        return Host_Error("CL_ParsePacketEntities: Error creating entity");
    }
    ...
    ent = (entitylist->vtbl->GetClientNetworkable)(u->m_nNewEntity);
    if ( !ent )
    {
        ...
        return Host_Error("CL_ParseDelta: invalid recv table for ent %d.\n", u->m_nNewEntity);
        ...
    }
}
```



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- The function `CL_CreateDLLEntity()` will create an entity if it doesn't exist (i.e `GetClientNetworkable` returns `NULL`)
- Eventually, the function `AddEntityAtSlot()` is called
- In the listing below, `EntityArray` points to another array in the `entitylist` object

```
int *CBaseEntityList::AddEntityAtSlot(unsigned int *EntityArray, IClientNetworkable *ent_object, int
index, int serial_num)
{
    unsigned int *object_ptr; // eax@1
    ...

    object_ptr = &EntityArray[4 * index + 1];
    *object_ptr = ent_object;
    if ( serial_num != -1 )
        EntityArray[4 * index + 2] = (unsigned int)serial_num;
    ...
}
```

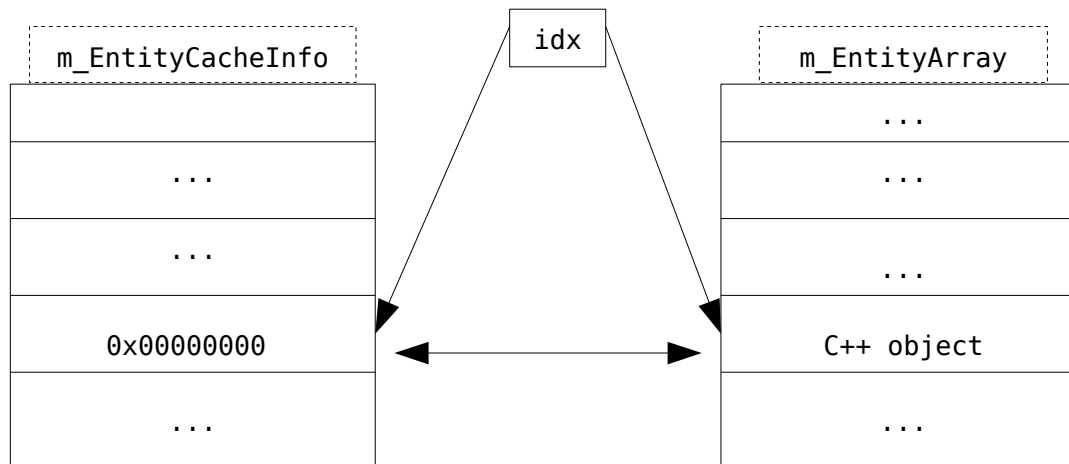


Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- We have two arrays: `m_EntityArray` and `m_EntityCacheInfo`
- We have one index
- We can simplify the accessing of the two arrays like so:
 - `m_SharedArrayBase[idx * 8 + 0x28] → m_EntityCacheInfo`
 - `m_SharedArrayBase[idx * 16 + 4] → m_EntityArray`
- If the value fetched from the `m_EntityCacheInfo` is NULL, we will write a C++ object in the `m_EntityArray` (do not forget integer wrap)



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- We now have an indexed write of a C++ object's pointer
- If we can read it back as a string, we can leak its vtable pointer and defeat ASLR
- How can we possibly do that ...
- Remember CVars ? :)
- Some Cvars are stored in the global data section of client.dll so we can corrupt them
- CVars have a `char * str_value` field, we can replace this field with a C++ object's pointer and query the Cvar through `svc_GetCvarValue`



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- Search algorithm to find good Cvar targets

```
for cvar in cvar_list:
    assert(cvar.inclientdll(cvar))
    offset = 0x80000000 + ((cvar.addr + off_to_str - m_ShareArrayBase) / 16)
    if m_ShareArrayBase[offset * 8 + 0x28] == 0:
        return (offset, cvar.name)
```

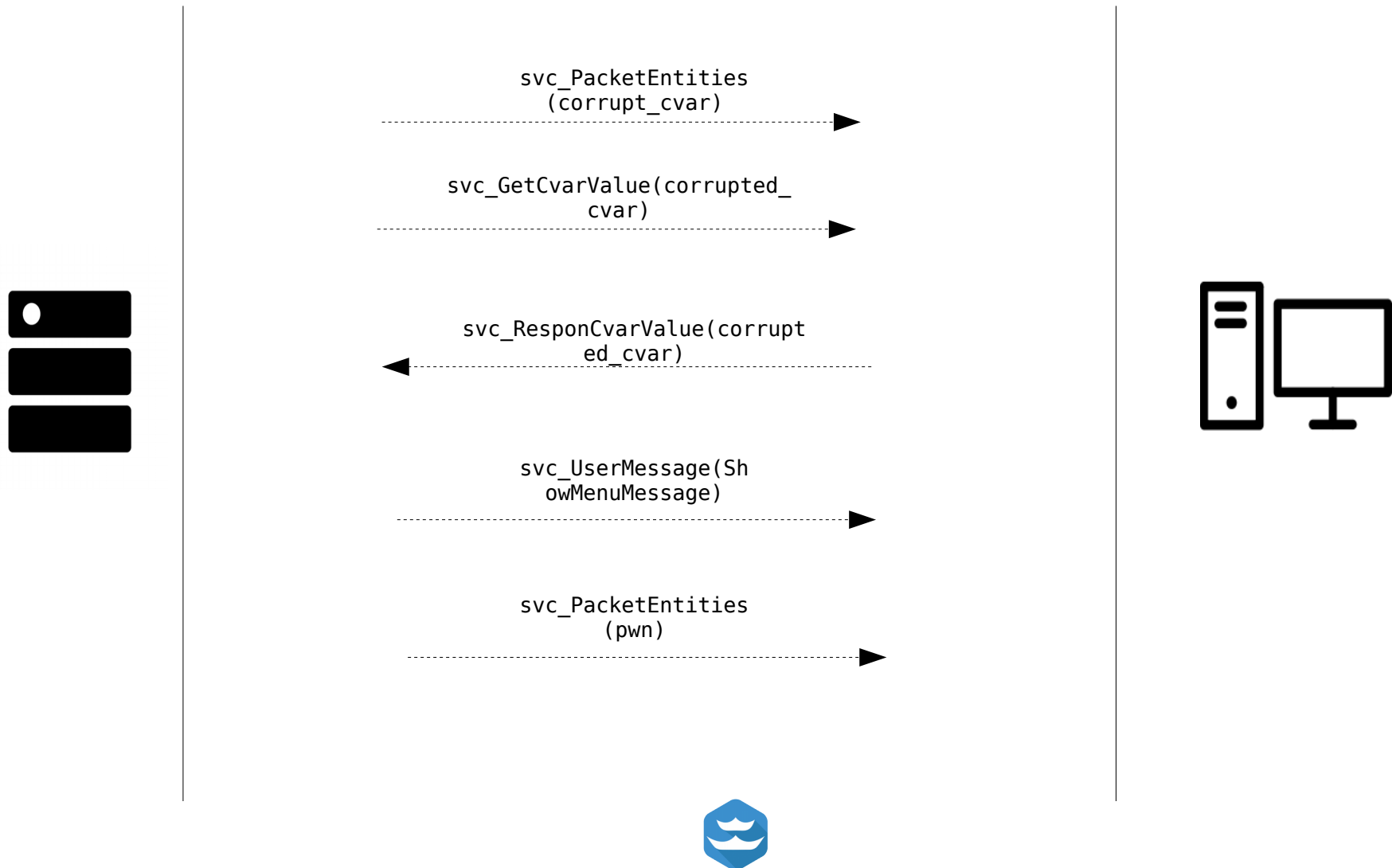


Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- Now attack plan looks like:



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- We got a problem though
- Due to the integer overflow ($\times 8$ vs $\times 16$), after writing the C++ object, the check below will fail
- The function `Host_Error` results in the client disconnecting from the host
- Once the client disconnects, user interaction is required to force it to connect again
- This is lame, can we do anything about it ?

```
int __cdecl CL_CopyNewEntity(CEntityReadInfo *u, int iClass, int iSerialNum)
{
    ...
    ent = entitylist->vtbl->GetClientNetworkable(u->m_nNewEntity);
    if ( !ent )
    {
        ...
        return Host_Error("CL_ParseDelta: invalid recv table for ent %d.\n", u->m_nNewEntity);
        ...
    }
}
```



Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- Remember ConCommands ? :)
- One particular command is of interest to us:
 - retry: Retry connection to last server.
- Obviously we can't send this command to the client after it disconnects...
- Send the command in the same packet as the svc_PacketEntities packet
- Due to a delay in command processing
- Command ends up still being processed after the client disconnects

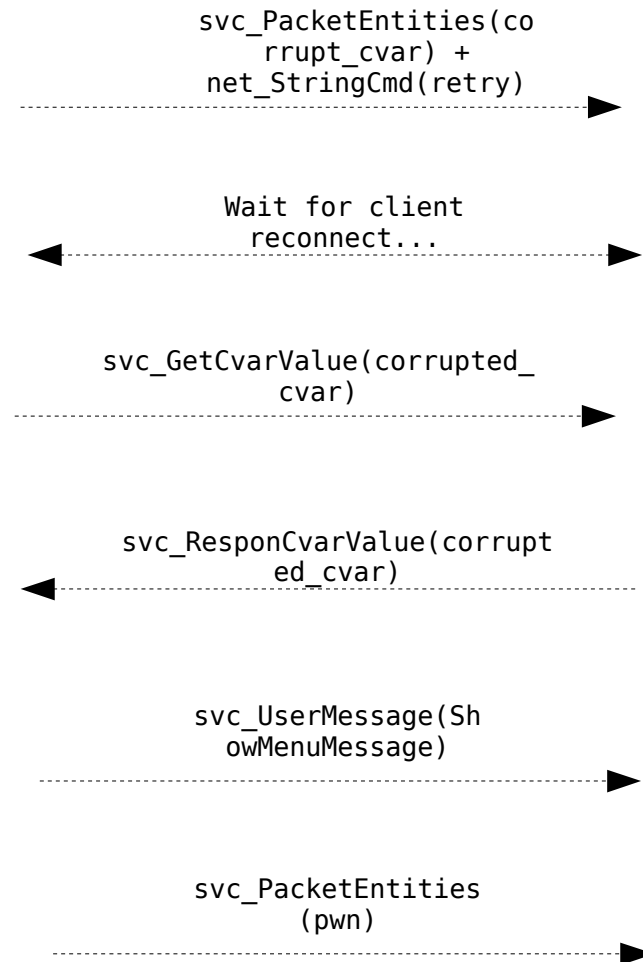
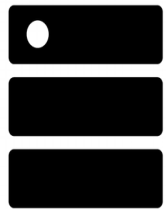


Hunting for Bugs



Signedness Issue in CL_CopyNewEntity() in Source Engine core

- Updated attack plan is:



Demo



Conclusions and Future Work



Conclusions and Future Work

- Game engines aren't safe
- **DO NOT** connect to random game servers
- Fuzzing
- Attacking the Servers instead of the Clients
- How about jailbreaking consoles through game engines ?



? 's



References

- <http://aluigi.altervista.org/>
- <https://developer.valvesoftware.com/wiki/>
- <https://forums.alliedmods.net/>
- [https://en.wikipedia.org/wiki/Source_\(game_engine\)](https://en.wikipedia.org/wiki/Source_(game_engine))
- <https://revuln.com>
- https://en.wikipedia.org/wiki/Game_engine

