Version Specific Dumps

- Season 31
- Season 30
- Season 29
- Season 27
- Season 25

Season 31

	captured at	game build	memory dump completeness	links	Comment
Generic	09.11.2026	R5pc_r5-10_J24_CL11700487_FSv31_0_2026_10_23_17_04	N/A	**No IDB Present**	

Anybrain Changes

A new convar named "**sv_player_input_experimental_framework**" has been added with this update. Its being checked inside some of the raw input telemetry functions we had already previously discussed. Once i finish some indirectly related work on my tooling, i will also update this post if i can collect evidence of its activation state.

Season 30

Type	captured at	game build	memory dump completeness	links	Comment
Generic	16.10.2026	R5pc_r5-301_J65_CL11630615_FSv30_1_2026_10_02_09_54	N/A	**No IDB Present**	

> **AC Migration:** The game is now protected by EA Javelin instead of EAC. Alongside it, you will find R5AC still being deployed and doing similar stuff that we uncovered back in the day. Additionally, client-side code is present for an integration of Anybrain, a solution advertised as an AI-based cheat detection. Although it should be mentioned that it wasn't active within my own limited tests.

> **Packer Removal (partly):** It seems like respawn is no longer applying dynamic memory protection onto the primary game executable, r5apex_dx12.exe. I'm unable to claim that everything related to their old Theia implementation is gone, as it was doing much more than just packing, for example it also hard stuff for hardware identification and broader anti-tamper functionality. This means you will be able to obtain a full memory dump without having to defeat a packer that periodically re-encodes stale pages.

Useful Codesites and Globals

Experiments System

Name	Address	Comment
R5::FetchExperimentsForProfile	%DUMP_BASE%+0x1AC7A10	Sends an HTTP/S request towards an endpoint URL "experimentation/v1/fetch-experiment-grouping". It attaches various parameters including Player/Nucleus-ID, and then receives a list of experiments and their individual states.

R5::ParseExperiments	%DUMP_BASE%+0x1AC8430	Parses the experiment list that was previously received with the previous function. It has a special handling case for "8-1-anyb-rollout", which if the experiment is determined as active and you're within the active treatment group, a ConVar named raw_input_client_send_enable is enabled.

Input Telemetry (Anybrain)

R5::ProcessInputTelemetry_Opaque_1	%DUMP_BASE%+0x1CBDD60	Processes some input telemetry related data, and conditionally writes it to a global ring buffer.
R5::ProcessInputTelemetry_Opaque_2	%DUMP_BASE%+0x1D830B0	
R5::InputTelemetrySender::1	%DUMP_BASE%+0x1D834D0	Used to send player input telemetry using standard CNetChan facilities. (SendNetMsg+SendDatagram)
R5::InputTelemetrySender::2	%DUMP_BASE%+0x1D82E00	Uses sub_1A36B90 to indirectly invoke R5::InputTelemetryDispatcher::1
R5::InputTelemetryDispatcher::1	%DUMP_BASE%+0x1D832A0	Used to send player input telemetry using standard CNetChan facilities. (SendNetMsg+SendDatagram)

Game Engine

CL_Move	%DUMP_BASE%+0x1AF4C30	
CL_SendMove	%DUMP_BASE%+0x1AF49E0	
C_ClientRenderable::SetupBones	%DUMP_BASE%+0x1FE6950	

R5AC

EnqueueViolation	%DUMP_BASE%+0x1D85A90	Appends a new anti-cheat telemetry message into the global queue, in many cases for a violation of some sort, but it hasn't been proved that it is only used for violations.

EnqueueOrSendViolationDirectly	%DUMP_BASE%+0x1D85BF0	Essentially something similar as the above EnqueueViolation function, with one difference being that it has a fast path: If your game client has an active network channel to a game server, it will attempt to send the anti-cheat telemetry message directly using engine API's like C_NetChan::SendNetMsg and C_NetChan::SendDatagram.). Otherwise it just calls into EnqueueViolation to do the usual queueing.
InitializeStackWalker	%DUMP_BASE%+0x1D83E40	Responsible for various things: • Initialization of imports for R5AC stack walker / control flow inspection • Initialization of the code execution whitelist. Its exact format is to be studied. • Sets a global variable of type u8 to 1 when initialization succeeds. The inlined stack walker code in game & engine functions then checks this alongside their other calculations for determining whether the stack walker should run. My understanding of this list is that it also is very likely a region-based whitelist and not per module, like you would initially think. So they kept that design choice and just switched it with a hash map. There seems to be some other changes too, including how the whitelist is built. In one occasion, it searches for a specific module in the LDR, loops through its regions and adds any section that is executable, non-discardable to its whitelist structure.

Useful Patches

Below is a list of convenient patches that you can apply, to enhance your experience within Apex. The best part is you're not introducing anything artifical into the mix. If you're a controller player

and already used to the feeling of aim assist, for example, you can just buff it directly. Note that this is probably no longer a golden ticket with the rise of Artifical Intelligence in EA's playbook.

Another positive aspect is that you don't necessarily need to patch anything to achieve much of these. For example, with the name hider, you can just manipulate a couple of global variables within Apex to make the lookup fail. For the aim assist stuff, you could reverse engineer their playlist variable system more deeply, and then figure out how to do it more stealthily.

Same thing applies for the deadzone check. Basically, inside a mouse related processing function at r5apex_dx12.exe, it just sets some global u64 which represents the last time your mouse has moved. You can log the values, see when they change, improve your model of the surrounding code and then find a way to achieve it in a much more subtle way. My approach saved me a lot of time, because for the most time I could afford changing whatever I wanted. This is subject to change but even if that's no longer sustainable, it's still valuable for things like narrowing down the amount of code you need to reverse engineer.

Patch Name	Address / Offset	Patch Bytes
Gamepad Assist Strength Changer	`0x20A83F0`	`48 31 C0 0F 57 C0 48 C7 C0 CD CC 4C 3F 66 0F 6E C0 C3` `moves 0.8f in hexadecimal representation into XMM0`
Mouse Inactivity Check	`0x20A7022`	`90 90 90 90 90 90` `just a bunchof NOPs`
Deadzone Check	`0x20A8753`	`EB`
Should Enable Aim Assist	`0x20A6FE0`	`B0 01 C3`
Input Enforcer 1	`0x21B1380`	`B0 01 C3`
Input Enforcer 2	`0x21B12E0`	`B0 01 C3` `when patching this and the first one, aim assist will work on mouse and keyboard. a controller needs to be attached nonetheless even if you can just play on MnK afterwards.`
Name Hider	`0x1A40391`	`E9 B2 00 00 00 90` `when patched, player names in your hud and world text will appear as Unknown:Unknown`

> EA's kernel driver is currently performing reads on various addresses within the game's code section. Whether that is for prevention or detection, i can't tell. But it would make sense for both use cases. During a short test session, it was most prominently a couple routines inside

> eaanticheat.sys accessing my target pages. Once I'm able to answer some of my open questions regarding this mechanism, I will update this book to reflect any new insights. You should expect them to have multiple integrity checks for something this obvious, but let's not overstate it too early.

Season 29

Type	captured at	game build	memory dump completeness	links	Comment
Generic	02.07.2026	R5pc_r5-291_J84_CL11262393_2026_07_02_15_15	51%	**No IDB Present**	***work in progress**

> **Possible integration of new cheat detection:** It seems like Apex has added some code that is part of an AI cheat detection solution. Currently, their approach seems to not deploy it for every player, but use an A/B testing system that Is externally controlled. It seems to be still in experimentation stage in Season 29, maybe they will roll it out globally later. This specific dump might not contain much besides strings initially (for the AI anti cheat related stuff), but you can manually fix up the dump to discover the code covering A/B testing implementation and their configuration schema for the whole thing.

EDIT: For locating the A/B rollout routine, check **%DUMP_BASE%+0x2CEE50.** Additionally, a wrapped function calling said routine is being assigned to a pointer within an object in the function at **%DUMP_BASE%+0x422BFD**. Below are some additional functions.

Name	Address	Comment
EXP_SendActiveExperiments	%DUMP_BASE%+0x2CF660	~
R5::FetchExperimentsForProfile	%DUMP_BASE%+0x2CE350	Sends an HTTP/S request of type POST towards an endpoint URL "experimentation/v1/fetch-experiment-grouping". It attaches various parameters including Player/Nucleus-ID, and more.
~	%DUMP_BASE%+0x57EE94	*work in progress
~	%DUMP_BASE%+0x5807E0	submit rawinput telemetry via C_NetChan::SendNetMessage

Plaintext Strings

> .rdata:00007FF7DEC95258 00000016 C anticheatMessageCount
> .rdata:00007FF7DECB3778 00000034 C
> anticheat.raw_input.match.send_buffer_overflow:1|c\n
> .rdata:00007FF7DECB37B0 0000002B C
> anticheat.raw_input.match.max_rate:%.2f|g\n
> .rdata:00007FF7DECB37E0 00000026 C
> anticheat.raw_input.match.rate:%ld|g\n
> .rdata:00007FF7DE97EA48 0000002C C c:\\depot\\r5-
> 291\\src\\security\\anybrain.pb.cc
> .rdata:00007FF7DE97DCD2 0000000E C anybrain.proto
> .rdata:00007FF7DE97DCE1 00000017 C \bAnybrain\"U\n\nPlayerData
> .rdata:00007FF7DE97DD0C 00000012 C .Anybrain.Metadata
> .rdata:00007FF7DE97DD2E 00000014 C .Anybrain.InputData\"
> .rdata:00007FF7DE97DD9E 00000014 C .Anybrain.ScreenInfo
> .rdata:00007FF7DE97DDCF 00000019 C .Anybrain.InputDeviceInfo
> .rdata:00007FF7DE97DDFF 00000022 C .Anybrain.ClientInfo\"\\\n\nScreenInfo
> .rdata:00007FF7DE97DEC9 00000020 C .Anybrain.MouseSens\"-\n\tMouseSens
> .rdata:00007FF7DE97DF36 00000019 C .Anybrain.OperatingSystem
> .rdata:00007FF7DE97DF66 00000015 C .Anybrain.Environment
> .rdata:00007FF7DE97DF92 00000014 C .Anybrain.FormFactor
> .rdata:00007FF7DE97E002 00000014 C .Anybrain.MouseInput
> .rdata:00007FF7DE97E02A 00000017 C .Anybrain.KeyboardInput
> .rdata:00007FF7DE97E052 00000014 C .Anybrain.TouchInput
> .rdata:00007FF7DE97E07C 0000001A C .Anybrain.ControllerInput\"
> .rdata:00007FF7DE97E0D6 0000001A C .Anybrain.MouseEventButton
> .rdata:00007FF7DE97E107 00000019 C .Anybrain.MouseEventMove\"
> .rdata:00007FF7DE97E14B 00000019 C .Anybrain.MouseButtonType
> .rdata:00007FF7DE97E1CE 00000012 C .Anybrain.EventRLE
> .rdata:00007FF7DE97E1ED 00000012 C .Anybrain.EventRLE
> .rdata:00007FF7DE97E20F 00000012 C .Anybrain.EventRLE
> .rdata:00007FF7DE97E235 00000012 C .Anybrain.EventRLE
> .rdata:00007FF7DE97E260 0000001E C .Anybrain.EventRLE\"+\n\bEventRLE
> .rdata:00007FF7DE97E2DD 00000028 C
> .Anybrain.KeyboardEvent\"J\n\rKeyboardEvent
> .rdata:00007FF7DE97E376 00000014 C .Anybrain.TouchEvent
> .rdata:00007FF7DE97E398 00000014 C .Anybrain.TouchEvent
> .rdata:00007FF7DE97E3BA 00000022 C .Anybrain.TouchEvent\"d\n\nTouchEvent
> .rdata:00007FF7DE97E493 0000001F C .Anybrain.ControllerEventButton
> .rdata:00007FF7DE97E4C2 00000025 C (\v2!.Anybrain.ControllerEventJoystick
> .rdata:00007FF7DE97E4F7 00000027 C (\v2
> .Anybrain.ControllerEventTrigger\"h\n
> .rdata:00007FF7DE97E54B 0000001E C .Anybrain.ControllerButtonType
> .rdata:00007FF7DE97E5AE 00000017 C .Anybrain.JoystickEvent

```
.rdata:00007FF7DE97E5D3 00000028 C
.Anybrain.JoystickEvent\"3\n\rJoystickEvent
.rdata:00007FF7DE97E647 00000016 C .Anybrain.TriggerEvent
.rdata:00007FF7DE97E66B 00000019 C .Anybrain.TriggerEvent\",\n
.rdata:00007FF7DE97EA78 0000000F C anybrain.proto
.rdata:00007FF7DE97F8F8 0000001A C Anybrain.Metadata.user_id
.rdata:00007FF7DE97F918 0000001D C Anybrain.Metadata.session_id
.rdata:00007FF7DE97F938 0000001D C Anybrain.Metadata.ip_address
.rdata:00007FF7DE97F958 00000025 C Anybrain.InputDeviceInfo.device_name
.rdata:00007FF7DE97F980 0000001D C Anybrain.InputDeviceInfo.vid
.rdata:00007FF7DE97F9A0 0000001D C Anybrain.InputDeviceInfo.pid
.rdata:00007FF7DE97F9C0 0000001E C Anybrain.ClientInfo.device_id
.rdata:00007FF7DE97F9E0 00000021 C Anybrain.ClientInfo.device_model
.rdata:00007FF7DE97FA08 0000001F C Anybrain.ClientInfo.user_agent
.rdata:00007FF7DE97FA28 0000001C C Anybrain.KeyboardEvent.code
.rdata:00007FF7DEC820F9 00000011 C 8-1-anyb-rollout
```

Season 27

Type	captured at	game build	memory dump completeness	links	Comment
Game/Engine	30.11.2025	R5pc_r5-270_J51_CL10241244_2025_11_19_18_56	71%	**No IDB Present**	

Season 25

Type	captured at	game build	memory dump completeness	links
AntiCheat	16.05.2025	R5pc_r5-250_J14_CL9041646_2025_04_25_16_44	60%	**No IDB Present**