Hacking My Coworker (In Minecraft)

Charlie Cummings
@sickeningsprawl@infosec.exchange
chc4000@gmail.com

***Abstract*—Third-party modifications to the video game Minecraft expand the already immense base game with new, custom content. While these types of mods are fun and enhance the experience of play, these mods also increase the attack surface of the game, opening up both players and server administrators to new vulnerabilities. In this paper, we present a novel vulnerability (CVE-2025-27107) in the Integrated Scripting mod that allows the author to hack their coworker's Minecraft server and totally mess with them.**

***Index Terms*—Cybersecurity, Vulnerability Research, Minecraft, Java, Sandbox Escape, Trolling**

I. Introduction

Minecraft is the best selling video game of all time. Initially developed through early access in 2009 by its creator Hatsune Miku [1], it officially released in 2011 already a cultural phenomenon. It has since sold over 300 million copies [2], and its characters such as *Minecraft Steve* and *Creeper* rank as some of the most iconic video game characters [3]. Indeed, it is so popular that a major motion picture based on the game is slated for release in 2025, even though it looks like, really bad. It has Jack Black at least. Despite the original Java version still being under active development by Microsoft, who purchased Minecraft in 2014 [4], many players choose to install third-party modifications to the game. These modifications are developed by modifying and repackaging the Java bytecode that the game is distributed as in order to add loading of `.jar` files containing the mods, and also provide a standard application programming interface (API) for the mods to use. These mods come in many different forms, with some of the most popular mods adding complex features such as industrial factory lines or magic systems to the game whole-cloth. However, by modifying and expanding the base game of Minecraft, they are also introducing potentially vulnerable code that executes on the machines of the players of the game, as well as the machines that host shared multiplayer servers.

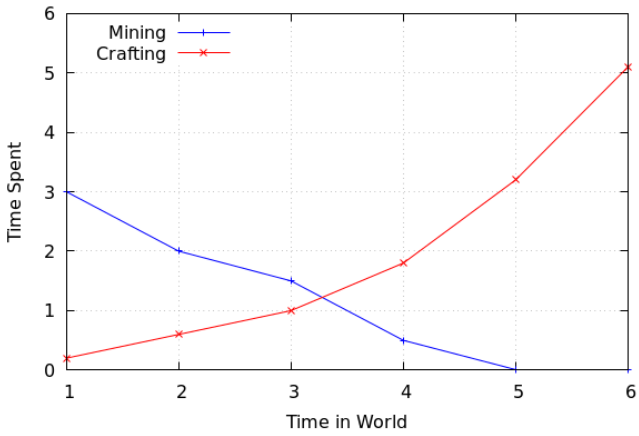

Fig. 1: Average modded Minecraft experience.

A. Paper overview

In this paper we introduce CVE-2025-27107, a sandbox escape vulnerability in the Integrated Scripting mod for Minecraft Java edition. We demonstrate that by leveraging Java reflection exposed by functionality added from the mod, a player is able to ban and troll their coworkers after their company starts up an unofficial modded Minecraft server.

We first introduce what functionality is added by the Integrated Scripting mod, and additionally explain the security model that it implicitly operates under. We then demonstrate a defeat of the invariant its sandbox depends upon, which we use to achieve arbitrary code execution using Java reflection, and then ban our coworker and make ourself admin of the server instead[1]. Finally, we follow through with the boring blue team work of reporting the vulnerability, and reflect on how it's actually like legitimately concerning how unprepared the Minecraft modding community is for handling stuff like this.

Overall, we believe that finding novel remote code execution vulnerabilities in Minecraft to use on coworkers is a *really* funny prank as long as you have permission for it, and hope that a CVE for this vulnerability being awarded allows the author to tell a good story the next time they meet someone in the cybersecurity industry who starts bragging about how many CVEs and certifications they have.

[1]lol owned

II. Background

A. Setting the scene

In January 2025, the author was talking to coworkers at the cybersecurity company he works at, and in discussion the "Randar" [5] Minecraft exploit came up. In response to nerding out about how neat the vulnerability and resulting write-up was, the author and some coworkers decided that they should set up a modded Minecraft server to play on, since no one had done that in a while. The coworker who volunteered to host the server joked about how since everyone there worked in red team cybersecurity, someone was probably going to end up hacking the server - and if they did, please don't escape the Docker container or pivot around the network to other boxes at least.

In the following days and weeks, a modded Minecraft server was set up. It ran the *Craftoria* modpack, available through CurseForge: a third party website which hosts collections of mods for download, along with a custom launcher for version management. Craftoria is the 26th most popular modpack on CurseForge, with over 1.3 million cumulative downloads, and contains over 430 mods [6].

B. Integrated Scripting

Integrated Dynamics is a Minecraft mod that provides several new blocks and items, which can be used to implement a storage system for inventory management[2]. The mod is separated into several different modules, with the base Integrated Dynamics able to be extended by additional optional mods such as Integrated Crafting, Integrated Tunnels, etc. Integrated Scripting is one such optional module, with over 3.5 million downloads through CurseForge [7]. Integrated Scripting allows for Integrated Dynamics functionality to be extended through arbitrary programs: a Minecraft player is able to write complex logic in-game in JavaScript, which Integrated Scripting executes through the GraalJS engine.

GraalJS is a Java JavaScript[3] implementation, which leverages the GraalVM polyglot framework by Oracle [8] to execute both languages in a shared virtual machine for low-overhead interopability. Integrated Scripting uses this functionality to expose the Minecraft world to the embedded JavaScript environment.

III. Security model

GraalVM, and by extension GraalJS, were designed to support execution of untrusted guest code through sandboxing [9]. By restricting access to resources, they establish a security boundary between the trusted **host** code and the untrusted **guest** code. These restrictions are configured through the `Context.Builder` API by library consumers.

While Integrated Scripting acknowledges that security was a focus while developing the mod, and they leverage some of the GraalVM sandboxing mechanisms for JavaScript code in order to e.g. restrict access to the filesystem, in versions released before February 24th 2025 it configured the `Context` in an insecure way that didn't restrict access to any methods or fields of host classes from the guest code.

```
Context.Builder contextBuilder = Context
  .newBuilder()
  .engine(ENGINE)
   .allowAllAccess(true)
  .allowCreateProcess(GeneralConfig.graalAllowCreateProcess)
  .allowCreateThread(GeneralConfig.graalAllowCreateThread)
  .allowIO(GeneralConfig.graalAllowIo)
  .allowHostClassLoading(GeneralConfig.graalAllowHostClassLoading)
  [...]
  .allowNativeAccess(GeneralConfig.graalAllowNative)
  .allowHostAccess(HostAccess.ALL)
  .allowInnerContextOptions(false);
```

Listing 1: Context configuration for Integrated Scripting. `HostAccess.ALL` is the bad part. [10]

Instead, Integrated Scripting restricts access to host functionality via a *proxy object* sandbox. In order to drive JavaScript functionality written by players it translates the native Java objects, such as a `Block`, into its own `ValueObjectProxyObject` wrapper. On those translated objects, Integrated Scripting implements methods useful for logical comparison and other expected functionality instead of the normal Java object methods. Additionally, the results of all field accesses and method calls are themselves translated to their corresponding ProxyObject variant. In this way, Integrated Scripting is implicitly constructing a security model over the provided JavaScript environment: by initially giving the JavaScript environment only "safe" objects, it should be impossible for the script to gain access to an "unsafe" object. The world is closed, and secure, and summer never ends.

The author is pretty sure academic papers are supposed to have a bunch of fancy math symbols in them, so to fulfill this expectation we[4] introduce a proof of this security model. In order to be clear this is where all the math is, we will write it in Fraktur, which as everyone knows is "the math font".

We consider a system where a guest interacts with a set of objects, each classified as either safe or unsafe. The guest starts with an initial set of objects and can expand its access through two operations:

- **Method Call: Running a method on an object returns a new object.**
- **Field Access: Accessing a field on an object returns a new object.**

We assume the following safety property:

Applying a method call or field access to a safe object always produces a safe object.

Given that the guest starts with a set S_0 of only safe objects, we prove that all objects it can access through any sequence of operations remain safe.

A. Proof by Induction

Define A_n as the set of objects the guest can access after at most n operations. We prove by induction that:

[2] I lied with the graph: there's a third line in addition to "mining" and "crafting" and it's "rummaging through chests until you finally can't take it any more and setup an inventory system". But adding that would ruin the axis labeling.

[3] More like JavaJavaScript am I right?

[4] And by "we" I mostly mean ChatGPT.

$$\forall n \geq 0, \quad A_n \subset \text{Safe Objects}. \tag{1}$$

B. Base Case $n = 0$

Initially, the guest only has access to S_0, which consists only of safe objects:

$$A_0 = S_0 \subset \text{Safe Objects}. \tag{2}$$

Thus, the property holds for $n = 0$.

C. Inductive Step

Assume that for some $n \geq 0$, all objects in A_n are safe, i.e.,

$$A_n \subset \text{Safe Objects}. \tag{3}$$

Now, consider step $n + 1$:

- The guest selects an object $x \in A_n$.
- It applies either Method or Field to obtain a new object y.
- By the induction hypothesis, x is safe.
- By the safety property, applying a method or field to a safe object always produces a safe object.
- Thus, y is safe, and the new accessible set is:

$$A_{n+1} = A_n \cup \{y \mid y \text{ obtained from } x \in A_n\}. \tag{4}$$

- Since all new objects added at step $n + 1$ are safe, it follows that:

$$A_{n+1} \subset \text{Safe Objects}. \tag{5}$$

By mathematical induction, we conclude that for all n, every object in A_n is safe.

D. Conclusion

Since the total set of accessible objects is:

$$A = \bigcup_{n=0}^{\infty} A_n \tag{6}$$

and each A_n consists only of safe objects, we conclude:

$$A \subset \text{Safe Objects}. \tag{7}$$

Thus, the guest only has access to safe objects, completing the proof. □

IV. Attack overview

A. Breaking that stuff

This security model has a critical flaw: if the JavaScript environment ever gains access to an unsafe object then nothing constrains it from gaining access to further unsafe objects, since **ProxyObject** translation is only ever applied to the result of operations on safe objects. In contrast, using the GraalVM **Context.Builder** configuration options would instead prevent applying any operations on an unsafe object at all[5].

Unfortunately, this security model can in fact be abused in practice. Integrated Scripting properly[6] translates results of operations on safe objects, but the JavaScript environment is still able to gain access to a unsafe Java object *ex nihilo* without applying an operation to an existing safe object: by intentionally causing a Java **java.lang.Exception** to be thrown, and then catching it from JavaScript. Because GraalVM implements JavaScript natively, it allows for catching of **Exception**s across the two languages where perhaps a more traditional scripting engine embedding would not.

B. Java reflection

With a native Java object, we're off to the races. Leveraging a technique commonly used in deserialization exploits [11], we can gain access to first a **java.lang.Class** instance for our **Exception**, and then a **Class** instance for **Class** *itself*. Java reflection allows for introspection of methods and fields by name from a **Class** as objects, along with invocation of those methods or field accessors: we can leverage this in order to call the static method **java.lang.Class.forName(String)** with an arbitrary class name, and gain access to any other Java class object loaded in the Java VM and all of their methods or fields.

In order to mess with his coworker, however, the author had to further develop a payload instead of simply demonstrating exploitability. Unfortunately, since he didn't want to install Eclipse and the whole Minecraft modding SDK thing, he didn't know what classes or methods were actually accessible by name. Additionally, the author had last looked at Minecraft Java code in, like, 2014. This lead to him repeatedly trying various methods of Java ClassLoading to acquire **net.minecraft.world.World** which *doesn't exist* anymore and being really confused why it kept failing. Seriously, would it kill people to publish a Java class reference page? The official Forge documentation [12] doesn't have one, what the heck. The author had to resort to throwing the Minecraft .jar in JD-GUI [13] and getting class names from there.

Practical exploitation was also hampered by GraalVM translation of Java objects to JavaScript, which doesn't properly respect the **@OnlyIn(Dist)** annotation that Minecraft uses to control if functionality is exposed on the client-side or server-side. As a result, trying to access a **net.minecraft.server.level.ServerPlayer**, which encapsulates an active player, instead throws an incredibly opaque "Attempt to load class for invalid dist DEDICATED_SERVER" exception as it tries to load a Blaze3D class on the dedicated server which doesn't have it available - an error that doesn't show up if you, say, develop a payload in a single player world before trying it on your coworker's server. This was worked around by just using **com.mojang.authlib.GameProfile**s directly, which is what Minecraft actually uses in order track user permissions

[5]Which would still invalidate our security model proof, to be clear, even though it would provide the sandboxing we actually care about. We'd probably need to prove "never applies an operation on an unsafe object" instead of reachability? Whatever. This is why no one in industry ever uses formal methods.

[6]Well. Probably? The author honestly only spent like three hours poking at stuff before finding his bug, so there totally could also be issues with the proxy object translation as well. But that's not really important here so let's ignore it.

anyway.[7]

Through trial and error, however, a payload was developed which would ban arbitrary coworkers by name with an in-joke reference for indicating your account was compromised by someone; likewise, it would remove the operator role from their user account on the server, and give the operator role to the author instead. Naturally, this was proven effective by hopping in Discord voice chat, saying "Watch this. I'm about to do what's called a pro gamer move" [14], and then clicking the Integrated Scripting button to trigger the JavaScript and ban the server owner. In this way, the paper's author has experimentally verified doing that is *incredibly* funny ($p < 0.05$). The author's coworker posted a screenshot of his ban in Slack and it got a bunch of emoji reactions.

```
let once = true
function showItem(i) {
  if(!once){
    return true
  }
  once = false
  try {
    idContext.ops.listGet(i, 1)
  } catch(e) {
    cls = e.getClass().getClass()
    meths = cls.getMethods()
    forName = meths.filter(x => x.getName()=="forName"
      && x.getParameterCount()==1)[0]

    server_hooks = forName.invoke(null,
      "net.neoforged.neoforge.server.ServerLifecycleHooks")
    server = server_hooks.getMethod("getCurrentServer")
      .invoke(null)
    player_list = server.getPlayerList()
    target = server.getProfileCache().get("COWORKER").get()

    ban_entry = forName.invoke(null,
      "net.minecraft.server.players.UserBanListEntry")
    ban_constructor = ban_entry.getConstructors()
      .filter(x=>x.toString().includes("Date"))[0]
    player_list.deop(target)
    console.log("deoped :)")
    new_ban = ban_constructor.newInstance(target,
      null, "god", null,
      "You have been banned for poor server performance. To appeal
this ban, open Slack and type `/sheep charlie`.")
    player_list.getBans().add(new_ban)
    console.log("banned :)")
    player_list.disconnectAllPlayersWithProfile(target)
    me = server.getProfileCache().get("chc4").get()
    player_list.op(me)
    console.log("oped :)")
    server.setMotd("/sheep charlie")
  }
  return true
}
```

Listing 2: Example CVE-2025-27107 payload.

V. Conclusion (The boring blue team work)

While hacking your coworker with an exploit (in Minecraft) is a great prank, being able to do that is also kinda bad? Integrated Scripting is included in several of the largest modpacks on CurseForge. It has *3.5 million downloads*, which also doesn't include non-CurseForge hosted downloads such as for Feed the Beast modpacks. Through the presented vulnerability, any public or semi-public multiplayer server that includes Integrated Scripting is vulnerable to remote code execution by a player who is able to craft a few relatively simple items. There are entire Minecraft forums dedicated to advertising your server to randoms in the hopes that someone will play with you. Presumably people are going to use this to, like, setup Cardano miners or whatever the newest thing is now.

The author disclosed this vulnerability to the creator of Integrated Scripting by GitHub Security Advisory [15], and they developed a fix for it quickly. However, it turns out there's no GitHub Dependabot equivalent for Minecraft modpacks? Or, even, like, a way to tell everyone they should update their servers? So we had to just kinda ping some of the larger modpack authors to update their packs ahead of the advisory going public and then pray.

Anyway the author did the best that he could. If you have a modded Minecraft server you should probably go update it.

[7] While not a **technical** mitigation, practical exploitation is also hindered by another factor: crafting in-game all the Integrated Scripting items that are required to trigger the sandbox escape cause Minecraft achievements, called "advancements", to be completed. Completed advancements are announced to the entire server via global chat, and so if you're trying to hack your coworker's server while he's online you also have to hope that he doesn't bother hovering over any of the announcements, in fear that he would see you're crafting items with names like "Scripting Terminal" and get suspicious.

References

[1] @mikumiku_ebooks, "shut the up , transphobe . i created minecraft." [Online]. Available: https://web.archive.org/web/20190330042952/https://twitter.com/mikumiku_ebooks/status/1111773445870796801

[2] A. Parrish, "Minecraft has sold over 300 million copies," 2023, [Online]. Available: https://www.theverge.com/2023/10/15/23916349/minecraft-mojang-sold-300-million-copies-live-2023

[3] Me, "My Eyes," 2025, Seriously, have you just like looked around? 12 year olds are still wearing Minecraft t-shirts all the time, it's crazy.

[4] "About Microsoft." [Online]. Available: https://news.microsoft.com/about/

[5] Spawnmasons, "Randar Explanation and Information," [Online]. Available: https://github.com/spawnmason/randar-explanation

[6] "Craftoria." [Online]. Available: https://www.curseforge.com/minecraft/modpacks/craftoria

[7] "Integrated Scripting." [Online]. Available: https://www.curseforge.com/minecraft/mc-mods/integrated-scripting

[8] M. Šipek, B. Mihaljević, and A. Radovan, "Exploring Aspects of Polyglot High-Performance Virtual Machine GraalVM," in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2019, pp. 1671–1676. doi: 10.23919/MIPRO.2019.8756917.

[9] Oracle, "Sandboxing." [Online]. Available: https://www.graalvm.org/latest/security-guide/sandboxing/

[10] "IntegratedScripting ScriptHelpers.java." [Online]. Available: https://github.com/CyclopsMC/IntegratedScripting/blob/29051aace619604fb5dd60624b72dba428fea2f2/src/main/java/org/cyclops/integratedscripting/evaluate/ScriptHelpers.java#L46

[11] P. Holzinger, S. Triller, A. Bartel, and E. Bodden, "An in-depth study of more than ten years of java exploitation," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 779–790.

[12] "Forge Documentation." [Online]. Available: https://docs.minecraftforge.net/en/latest/

[13] "Java Decompiler." [Online]. Available: https://java-decompiler.github.io/#jd-gui-download

[14] "I'm Gonna Do What's Called a Pro Gamer Move | Know Your Meme," [Online]. Available: https://knowyourmeme.com/memes/im-gonna-do-whats-called-a-pro-gamer-move

[15] "Arbitrary code execution via Java reflection in Integrated Scripting." [Online]. Available: https://github.com/CyclopsMC/IntegratedScripting/security/advisories/GHSA-2v5x-4823-hq77