NEACȘU-MICLEA LIVIU-ȘTEFAN (235)
CONDOR ANDRADA (232)

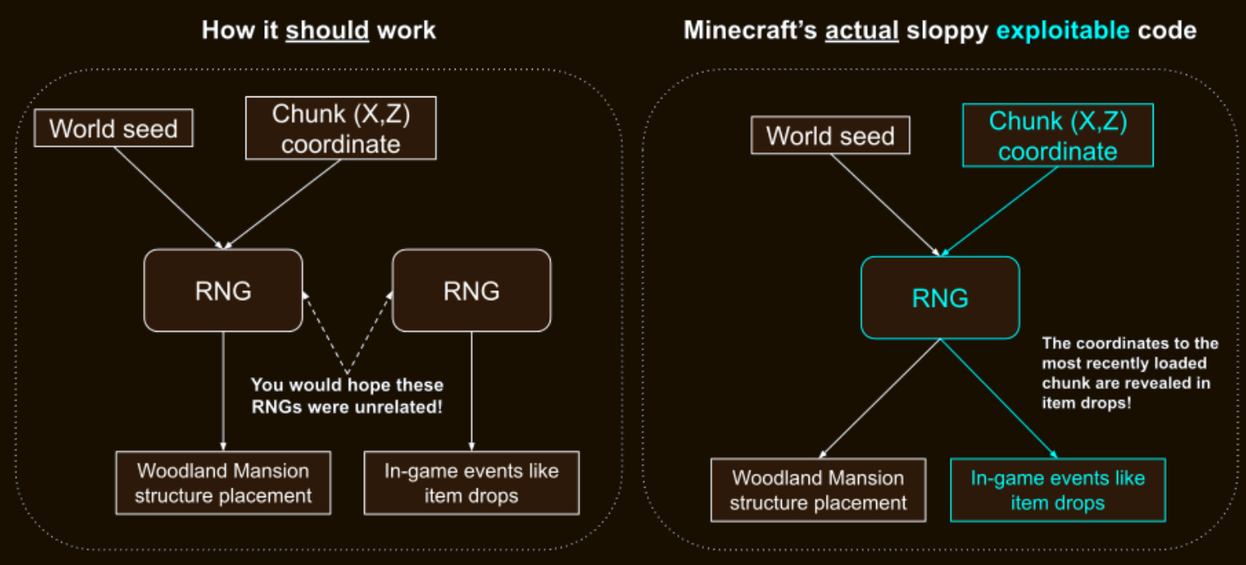
It was in plain sight

RANDAR EXPLOIT

RNG bug used to track players activity

THE ISSUE

For a good period in Minecraft's history, from Beta 1.8 to 1.12, the same random generator was dictating two independent factors in the game: the position of generated structures and small details like item drops or entities spawn.



HISTORY

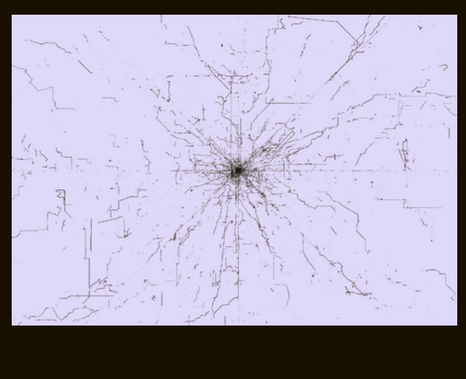
When Notch first added structures to the game in Beta 1.8 (2011), he accidentally reused an RNG that's supposed to be unpredictable in order to place Villages in the world. Ever since then, until 1.13, this sloppy code has caused world generation to influence nearly all other supposedly random events in the game. It took until around May 2018, for Earthcomputer and friends to discover this mistake, realizing that chunk loads affect the game's RNG in an observable way.

THE EXPLOIT

People have discovered that the whole process was reversible. Knowing the position of a dropped item, it was possible to find a subsequence of (**pseudo**)random generated output and hence determine the seed due to the way Java.util.Random works.

```
public float nextFloat() {
   this.seed = (this.seed * multiplier + addend) % modulus;
   int randomInteger = (int) (this.seed >> 24);
   return randomInteger / ((float) (1 << 24));
}
```

A malicious user would use a bot account on the server (its location does not matter) which drops items at "random coordinates". By using a "lattice method" mechanism to track down the source seed of the randomness, it finds out the location of the last loaded chunk. The collected data is used to build a heatmap of player activity, which provides valuable in-game information about players bases and positions.



A notable consequence of this exploit has happened on the oldest anarchy server in Minecraft, where a group of hackers (the SpawnMasons) have used Randar to fight players who were selling in-game items for real world money.

HOW IT WORKS

Whenever a players enters a chunk, the server checks if certain structures are being generated in that chunk. Due to sloppy code in Minecraft, the global World.rand has its seed reset to a function of the chunk coordinates, in order to check where a nearby Woodland Mansion should be (and whether it's this chunk in particular).

```
// (chunkX,chunkZ) is being loaded, and this function checks if it should generate a Woodland Mansion
protected boolean canSpawnStructureAtCoords(int chunkX, int chunkZ) {
   // divide by 80, rounding down, to determine which "Woodland region" (my made up term) we're considering
   int woodlandRegionX = Math.floorDiv(chunkX, 80);
   int woodlandRegionZ = Math.floorDiv(chunkZ, 80);
   // seed the random number generator deterministically in a way that's unique to this Woodland region
   Random random = this.world.setRandomSeed(woodlandRegionX, woodlandRegionZ, 10387319);
   // pick which chunk within this region will get the Woodland Mansion
   int woodlandChunkX = woodlandRegionX * 80 + (random.nextInt(60) + random.nextInt(60)) / 2;
   int woodlandChunkZ = woodlandRegionZ * 80 + (random.nextInt(60) + random.nextInt(60)) / 2;
   // but is it *this* chunk, that we're loading right now?
   if (chunkX == woodlandChunkX && chunkZ == woodlandChunkZ) {
      // and, is this chunk in a biome that allows Woodland Mansions? (e.g. roofed forest)
      if (this.world.getBiomeProvider().areBiomesViable(chunkX * 16 + 8, chunkZ * 16 + 8, 32, ALLOWED_BIOMES))
         return true;
   }
   return false;
}
```

```
/**
 * Spawns the given ItemStack as an EntityItem into the World at the given position
 */
public static void spawnAsEntity(World world, BlockPos pos, ItemStack stack) {
   double xWithinBlock = world.rand.nextFloat() * 0.5F + 0.25D;
   double yWithinBlock = world.rand.nextFloat() * 0.5F + 0.25D;
   double zWithinBlock = world.rand.nextFloat() * 0.5F + 0.25D;
   EntityItem entityitem = new EntityItem(world, pos.getX() + xWithinBlock,
                                          pos.getY() + yWithinBlock,
                                          pos.getZ() + zWithinBlock, stack);
   world.spawnEntity(entityitem);
}
```

The game does this checkup **every time a chunk is loaded**. The world seed was consequently used in other calculations, such as deciding where to place dropped item stacks.

HOW IT ENDED

The bug was fixed in 2018 when version 1.13 changed the world generation algorithm and also fixed the Randar bug. However, there are still Minecraft servers running on 1.12 or older versions which are vulnerable to the exploit. A native RNG patch does not protect against a possible attack. Even new version servers that were vulnerable in the past are still affected by the effects of the already collected information.