Exploitation of RNG Seed Predictability in RPG Video Games

Muhammad Ashkar - 13524007
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: ashkar.second@gmail.com , 13524007@std.stei.itb.ac.id

***Abstract*—Random Number Generators (RNGs) are important parts of video games, especially in role-playing games (RPGs). RNGs introduce randomness aspect to the game, which makes the game become unpredictable and more enjoyable. Something will be different every time you play the game. RNGs make games gives different experience every time you play it. In RPGs, RNGs have a role to random the chance of item drops, random enemy movements and actions, or even random the world where you play itself. Sometimes as you play the game, you will notice some random sequence of events always happen in such a pattern. When you know the pattern, you can start abuse it as you want. This is known as RNGs exploitation in video games.**

Keywords—RNG; Video Game; RPG; Exploit

I. Introduction

Random Number Generators or RNGs for short has a very important role in video games. It makes the game more interesting since it makes the games not monotone and not boring. Something always felt different in every games since there is a randomness aspect. This makes the games become more unpredictable and enjoyable.

Like its name, RNGs are used to make something random in the game, such as item drop, enemy movement, gacha system, or even make a random world from a set of object. Without this, the games will felt so boring because player already know what will happen in every run and will eventually memorize all contents of the games.

Nowadays, RNGs are found in almost every games, especially at Role-Play Games (RPGs). RPGs are the genre of the games where player can control a character to do a certain things. RPGs are famous because there is infinite possibility player can do with his character. In some RPGs, player can get an item by doing a certain things such as completing quest, opening treasure chest, and defeating an enemy. Sometimes the games will give you specific item for doing a certain things. But after you get that item, maybe you won't do that action again because you already have the item. What if we add several items that can be gotten from that action, with each item having its own chance to drop. This will make player more enthusiast while doing a quest or fighting enemies and maybe will redo it to get all items from those actions.

Some items has a extremely low drop rates causing players to spend dozens or even hundreds of hours only to get that item. These kind of item usually gives player a powerful effect in the game. Of course everyone want this kind of items, some people even willing to pay a real money only to get this kind of items. Some player want it to make his character better in game, some other want it for collection or even to sell it to another player.

This is also what makes RPGs fun to play even after thousands hours of playtimes. This unpredictability is very addictive like gambling. The feeling of getting that extremely rare item will make you want to play it even more and more. This makes people forget about time and spend all their time in the RPGs. This explain how impactful RNGs are in RPGs.

Many people on internet make a guide how to get that extremely rare item or how to make that rare enemy spawns. This guide usually tells you the fastest way to do the actions that has a chance to make those things happen. For example an extremely rare item can be dropped by specific enemy, the guide will tell you how to find those enemy, how to defeat it, and the fastest way to do it. But as you explore more guide, there will be a guide that tells you to do things that are not even related to the way to get that item. For example the guides tells you to kill monster A until it drops item X 5 times in a row, and then you kill monster B, then you will always got that very rare items from the first monster B you killed. Surprisingly a few of those weird guides sometimes work. This things has something to do with how RNGs work.

II. Theoretical Foundation

A. RNG

Random Number Generator (RNG) is an algorithm that produces a random number that can't be predicted. An output from this algorithm should be completely random so no one can guess it. RNG are fundamental concepts in many disciplines, including cryptography, statistic, video games, and probabilistic algorithms.

RNG is divided into two different types, True Random Number Generator (TRNG), and Pseudo-Random Number Generator (PRNG). Like its name, TRNG generates a completely random number with a completely random pattern. TRNG generates a random number based on physical processes,

such as noise, electric state of the machine, radioactive decay, and many others. This kind of RNG rarely used because it needs a special hardware to use and take more time to generate a number compared to PRNG.

In other side, PRNG generates a number that looks like a random, but if we observe it more, it has some kind pattern. This pattern occurs because PRNG uses some basis to start a randomizing process. This basis more commonly known as seed. From that seed, PRNG will start to randomize it using some kind of algorithm. The next time PRNG produce a number, the seed will be based on the random number generated from the previous PRNG, this what makes it has some kind of pattern.

B. PRNG Using Mathematical Modulo

The commonly used PRNG method is Linear Congruential Generator (LCG). This Pseudo-Random Number Generator use mathematical modulo to determine the output. The formula of this method is:

$$X_{n+1} = aX_n + c \bmod m$$

where X_{n+1} as an output number, X_n as previous random number, a as multiplier, c as increment, and m as modulus.

The value a, c, and m must be chosen carefully to ensure the RNGs are not easy to guess. The value of a, c, and m also must be chosen to make the RNG has a long period to repeat. If not chosen carefully, PRNG may generate a short repeating sequences of random number that can be predicted easily and fail to simulate the real random. A good set of a, c, and m can ensure the RNG will repeat in a long period of time. Good set of a, c, and m can be determined by using some theorem.:

m prime, c = 0

Lehmer RNG construction can be used for this case. Lehmer RNG can be viewed as a particular case of the linear congruential generator with c=0, it is a special case that implies certain restrictions and properties. In particular, for the Lehmer RNG, the initial seed X_0 must be coprime to the modulus m, which is not required for LCGs in general. The choice of the modulus m and the multiplier a is also more restrictive for the Lehmer RNG. In contrast to LCG, the maximum period of the Lehmer RNG equals $m - 1$, and it is such when m is prime and a is a primitive root modulo m.

m a power of 2, c = 0

This form has maximal period $m/4$, achieved if $a \equiv \pm 3 \pmod 8$ and the initial state X_0 is odd. Even in this best case, the low three bits of X alternate between two values and thus only contribute one bit to the state. X is always odd (the lowest-order bit never changes), and only one of the next two bits ever changes. If $a \equiv +3$, X alternates $\pm 1 \leftrightarrow \pm 3$, while if $a \equiv -3$, X alternates $\pm 1 \leftrightarrow \mp 3$ (all modulo 8). It can be shown that this form is equivalent to a generator with modulus $m/4$ and $c \neq 0$.

m a power of 2, $c \neq 0$

In this case, we use Hull-Dobell Theorem. According to Hull-Dobell Theorem, when $c \neq 0$, correctly chosen parameters allow a period equal to m, for all seed values. This will occur if and only if:

1. m and c are coprime,
2. a - 1 is divisible by all prime factors of m,
3. a - 1 is divisible by 4 if m is divisible by 4.

Although the Hull–Dobell theorem provides maximum period, it is not sufficient to guarantee a *good* generator. For example, it is desirable for $a - 1$ to not be any more divisible by prime factors of m than necessary. If m is a power of 2, then a - 1 should be divisible by 4 but not divisible by 8.

The seed has an important role in determining the sequence of numbers produced that will be produced by PRNG. With the seed, the entire sequence of outputs can be predicted. This is the problem in programs that initialize the seed with a fixed value. PRNG can easily be exploited by knowing the seed and formula used in programs. It can be predicted by replicating a program and using the same seed and PRNG formula.

PRNG using LCG is simple and fast, but not secure enough for cryptography. Attacker can observe the output from PRNG and use a program to know the algorithm and get the formula that used for PRNG. With a formula, attacker can reverse-engineer PRNG and get the seed used in PRNG. With seed and formula, attacker can predict the next value generated by PRNG.

Because of this reason, many modern system already used more advance PRNG algorithm, such as Mersenne Twister. Advance PRNG has a longer period and better random result, but still using the same concept with seed. This PRNG still can be exploited, but its harder to do since the period is longer and there's too much output to observe.

C. Relevant Number Theory Concepts

PRNGs, mostly based on modular arithmetic. Likes Linear Congruential Generator (LCG) that are rooted on number theory. These mathematical foundations is important in order to understand how PRNGs works. There are the concepts that are most relevant with the PRNGs, such as modular arithmetic, relatively prime numbers, and multiplicative inverse.

Modular Arithmetic

Modular arithmetic is a system of arithmetic for integers where the number will go back to 0 when it reaches a certain value called modulus. Modulo also known as the remainder of the value divided by modulus. If a and b give the same remainder when divided by m, it can be written:

$$a \equiv b \pmod m$$

In PRNG, this system make sure the random number will always be less than m.

Relatively Prime Numbers

In LCG, to reach the maximum period of repeating, certain conditions must be met. One of these conditions is some variables in LCG formula must be relatively prime to other variables. Relative prime number is a set of numbers, which its Greatest Common Divisor (GCD) is 1:

$$\gcd(a, b) = 1$$

Multiplicative Inverse

Multiplicative inverse is also important in analyzing PRNGs. Sometimes, it is possible to reverse the LCG formula to recover its previous random number. This reversibility is a reason why PRNGs are not secure enough for cryptography.

D. Seed

In PRNG, seed is the basis of randomizing. Seed has a very crucial role in PRNG and sometimes can even determine the PRNG pattern will be hard to be predicted or not. In some old or poorly developed games, seed can very simple such as seed that based on the seconds after the game is started or combination of text that is inputted by player such as name and other player identity. This seed will be recognized very easily by player, then player can start predict what will happen based on what already happened with the same seed before.

Good seed must be a set of number that can't be guessed and always change every times the program runs. There's many way to determine the seed, and the way to determine it shouldn't be able to be manipulated easily. There's an example of how seed generated:

System Time

This method using system time as a seed. The seed is set to timestamp such as millisecond or microsecond of the system when the seed is generated. This method is easy to implement and has many variety, but this method is weak, because it's easy to guess the seed by approximating the time of seed generation.

User Input

In this method, user input such as mouse movement and keyboard keystrokes will be used to determine the seed. This method is very hard to be predicted, since human can't do the same action with the perfect detail twice. This method has a weakness because it depends on user behavior and has a limited context.

Fixed Seed

This method use the exact same seed every time the program starts. This method only used for debugging and program testing. Fixed seed can always be predicted because it has no randomness aspect in repeated use.

Operation System

This method use some random number from system as a seed. This seed in unpredictable and suitable for cryptography. This method draw a seed from data on the operation system. This method has a slower process compared to others, but safe.

Hybrid

Mixing two or more methods is a good idea to generate seed, since it will be very hard to predict or even unpredictable. You can mix system time with operation system, or fixed seed with system time to make a random seed

E. RNG in Video Games

Role-playing Games (RPGs) has a lot of RNG usage in its program. Most RPGs rely on RNG too much. RNG used in many aspect, such as determining what will enemy do from a set of actions, will the item drops or not based on its chance, will player do more damage based on critical chance, and what loot player will get from opening treasure chest. Some games even determine what the world will looks like based on algorithm that use RNG to create a world from a set of objects that has already been selected based on the previous object selected.

In video games, we need a fast code to ensure games run smoothly. The fastest RNGs look like the best solution for it, but in fact the fastest not always the best. It depends on what RNG are used for. If it only used to random something that is not important in game, such as critical damage chance or enemy movement, LCG might be the best solution for it. But how if it is about opening a very hard to get treasure chest that can only be opened once a week? LCG might not be the best solution for it. We need a better RNG system to prevent pattern predictability.

III. RNG EXPLOITATION

Many RPGs out there have a simple RNG system. This RNG system is mostly found on an old RPG. This kind of RNG system can be easily exploited by knowing how RNG works. Players can exploit in with or without an external tool, based on how complex the RNGs are.

How does RNG exploitation work? Basically, you need to know the seed, the RNG algorithm or formula, and what will happen from that RNG output. With the seed and algorithm or formula, you can determine what output you will get from RNG. By knowing what RNG does, you can predict what will happen in the game.

The most common PRNGs, LCGs, use mathematical modulo to determine the output. By multiplying and incrementing the seed with a certain number, then use modulo to that number, you will get the output of the PRNGs. To do this, you need to know the

$$X_{n+1} = aX_n + c \bmod m$$

formula for LCGs system.

Not every RNG exploitation method needs knowledge of how RNGs work. There is a method that can be used only by recognizing patterns. Human brain can memorize some patterns that frequently occur.

A. Recognizing Repeating Pattern

The easiest way to do this is by observing the RNG patterns. There is a rare case in game when you notice some sequence of things that are based on RNGs most likely happen in that order. Then, by knowing that order, you can abuse it every time the pattern occurs to get the rare item that has this pattern.

For example, in RPGs you go to certain area and go killing monster, but there's a low chance for a golden monster to spawn every hour. The gold monster has a very low chance of dropping a golden weapon which is very powerful. There might be a pattern like when you got an item A from normal monster 3 times in a row and then go open a treasure chest and got a sword from that treasure chest, then if you kill golden monster after that, you will got golden weapon.

This can be exploited by killing the golden monster only after you found that pattern. Since golden monster itself are very rare and takes a very long time to spawn, that pattern might occur more frequently than the golden monster itself, so players exploit it to get the golden weapon faster.

Or in other cases, when the world of RPGs is always randomized every time you enter the game, and most of the things in that world have already been determined since the world generation, such as the item inside the chest, enemy spawn, enemy location, and item dropped by enemy. So, when the world generated the location of rare monster always the same if the seed is same.

Sometimes, there will be a pattern like when there are 5 stones facing south and 4 stones facing west in the starting area, the rare monster always spawns in the specific location. By recognizing this pattern, you can always find that rare monsters fast when this pattern occurs.

This rarely happens because if this happens, the seed or the formula of that RNGs is bad. The same pattern will always repeat in a short time, which means the RNG formula has a short period. This can be improved by changing the seed and formula to make its period longer and has a harder pattern.

B. Seed Manipulation

PRNGs always randomize based on its seed. By knowing how seed generation works in games, you can manipulate seed to get the exact same result every time. This method is rather hard to do, because you need to know how seed is determined and what will happen if you do a thing in that seed. This method is normally used in the game where the world is determined by the seed.

For example, in the game where the world is determined by the seed, and it reset every time you enter the game. You want to find and kill a lot of very rare monster to get a powerful weapon. By knowing how seed generated, you can manipulate the seed, so when you got a seed that generates a world with a lot of this very rare monster, you want to replicate it by manipulating seed generation process to make a seed that exactly same as the seed where you find a lot of very rare monsters.

This method is almost impossible to do without any tool to check the seed and generate the seed to be exactly same. But, in some games, you can dive into the game program to find a seed and then change it to the seed you want. With a proper tools, changing seed directly is easier than generating a specific seed.

C. RNG State Manipulation

With a proper tools, you can even manipulate the RNG state in runtime. There are many software out there that you can use to do this. Of course doing this need a knowledge of how RNG in that game works. In many games, RNG state is updated every time a random event is triggered. By manipulating this states, you can manipulate it so when you do certain action that use RNG, you want to make RNG produce a number that will end up as an output that you want.

In offline games, there's many software that can change the RNG state itself directly. So, you just choose a number you want to change and then enter the number you want it to be. But this is not easy to do, because you don't know which number you should change.

D. Reversing the RNG

In the expert hands, RNG can be reversed by knowing a few outputs from the RNG and knowing the algorithm being used in the program. For example, if you know the program using LCG, if the output and modulus are known, it's possible to calculate other variables to make a full formula. With a full formula and the basis of the current random number generation process, you can count the next RNG output with 100% accuracy.

This method needs a knowledge of how PRNG works, but this is the best way to predict the output of RNGs. When you know the output of the RNGs, you need to find out what those specific outputs will affect the game. It can either affect the background event that does almost nothing to the game or a crucial part such as item drops. You need to find it out.

E. Tools and Automation

In these modern days, there's many tools that can automatically predict the output of the RNGs. Some even can simulate the RNGs in real time. These tools monitor everything in the game that changes RNG state. With the data it got from monitoring every event in the game, these tools will start making a prediction for the next RNGs output.

The tools used in method are very complicated to make, but using it is very easy. You just need to learn how to use the tools and then use it in games. There are many tutorials on how to use those tools if you want to use it. The downside of this method is

the tools usually use a lot of resources from your device and sometimes make your game laggy.

In this AI era, automation can be done by AI which is very powerful. By using machine learning pattern recognition, you can predict the future values. You can also use AI to observe memory or anything that is related to RNGs. AI is powerful but also has an expensive cost.

IV. Prevention Technique

RNG exploitation is considered as cheating in video games, since you do something that you are supposed not to do. So, game developers tried everything to prevent this action. Doing RNG exploitation in online games may result in permanent ban. Modern game has many solutions to deal with this.

RNG in game can be better by using Cryptographically Secure Pseudo-Random Number Generator (CSPRNG). CSPRNGs, such as those based on SHA-256 and Fortuna are more resistant to reverse engineering because they rely on complex internal states. These RNGs may work slower than LCGs but provide better security from RNG exploitation.

RNG exploitation can also be prevented by using server-sided RNG. This way, it is impossible to check the state of RNG without hacking the server. This technique is already used in most games that rely heavily on RNG, such as games that have reward-sensitive system or gacha system. To ensure gacha result is not manipulated, it uses server-side RNG.

In offline games, there are no very strong solutions to prevent RNG exploitation. The best thing game developers can do is by making the seed and the algorithm is hard to guess. But since every single code in offline game must be run on local machine, users can always use external tools to check the machine state and manipulate the RNGs.

To make seeds harder to guess, seed must be dynamic and entropic. To generate seeds like this combining a normal seed generating method can be a solution. By generating a seed that is based on more than one thing, it can be very hard to manipulate, such as combining system uptime method and player input method.

Seed rotation method can also be used to make RNG exploitation harder. By changing a seed in every random period of times, guessing the seed can be harder to do. The period for the seed to change is also determined by RNGs. This seed rotation can also be made to sometimes skip some value so finding a formula becomes harder.

V. Conclusion

Random Number Generators (RNGs) has a vital role to make video games, especially role-playing games (RPGs) give a better experience by introducing a randomness aspect. However, RNGs sometimes become predictable. This is because RNGs not always a true random number generators (TRNGs), usually this is a pseudo-random number generators (PRNGs).

PRNGs with a poor seed or weak algorithm like Linear Congruential Generators (LCGs) are vulnerable to exploitation. Players can abuse repeating patterns, manipulate seeds, modify RNG states, or even reverse-engineer the RNG to get a very good reward without too much effort, especially in games that have reward-sensitive system that determined by PRNGs.

While these exploits are fun for some people, they ruin the game balance and are illegal to do. Game developers must come with a solution to prevent this such as Cryptographically Secure Pseudo-Random Number Generators (CSPRNGs) or server-sided RNGs. Understanding the mathematics theory and internal workings behind RNGs is a key to ensure RNGs in games work as intended and not easy to exploit.

References

[1] R. Payne, “Random number generators,” *Electronics Australia*, vol. 57, no. 3, pp. 85–89, Mar. 1992. [Online]. Available: https://www.firstpr.com.au/dsp/rand31/p85-payne.pdf

[2] W. R. Gilks, “Lecture 4: Random number generators,” Cornell University, Bioinformatics Course. Available: http://chagall.med.cornell.edu/BioinfoCourse/PDFs/Lecture4/random_number_generator.pdf

[3] https://www.fintools.com/wp-content/uploads/2012/02/RandomNumberGenerators.pdf

[4] S. Park and K. Miller, “Random number generators: Good ones are hard to find,” *Communications of the ACM*, vol. 31, no. 10, pp. 1192–1201, 1988. Available: https://www.researchgate.net/publication/220420979

[5] J. Spalding, “Generative art and randomness,” *Art 108, San José State University*, 2012. Available: https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1006&context=art108

[6] NCC Group, “Randomness,” Available: https://www.nccgroup.com/media/5ockoyyv/_randomness.pdf

[7] D. Halprin and E. W. Felten, “Private browsing: Great idea, but it doesn’t work,” in *Proc. 5th Symposium on Usable Privacy and Security (SOUPS)*, Mountain View, CA, USA, Jul. 2009, pp. 1–8. Available: https://cups.cs.cmu.edu/soups/2009/proceedings/a12-halprin.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 18 Juni 2025

Muhammad Ashkar - 13524007