Joe Grand's RoboForm Password Regeneration Technical Description

May 24, 2024

https://grandideastudio.com/portfolio/security/roboform-password-regeneration/
https://www.youtube.com/watch?v=o5IySpAkThg

Note: Our work has been tested on RoboForm 7.9.0 (released June 26, 2013) and all addresses/data described below is for that version. We believe the problem exists for earlier versions, as well, and for subsequent versions until RoboForm 7.9.14 (released June 10, 2015). According to the RoboForm changelog (https://www.roboform.com/news-windows), this version includes a change of "Password Generator: increase randomness of generated passwords."

While RoboForm's passwords appear to be randomly generated, they are actually deterministic based on the current system time (in seconds since January 1, 1970) when the "Generate New" button was pressed.

Using Ghidra, we reverse engineered passwordgenerator.exe and roboform.dll. Deep within the RunPassGen function within roboform.dll, we found a function at address 0x105e5390 that appeared to be the core password generator function. Within that function, at address 0x105e53e7, there is a CALL to __time64 (a Microsoft-specific function in the C runtime library that is used to obtain the current time in seconds elapsed since January 1, 1970). Only a few instructions later at 0x105e53f7 there is a call to _srand (which sets the starting point for the rand() function that produces pseudo-random integers later in the password generation process). As noted below, there is a bit of manipulation to that value before it is passed into _srand, but it's irrelevant to our attack. The rest of the password generation functionality all relies on the value passed to _srand at this exact point. So, if we can replay the seed value passed to _srand regardless of any pre- or post-manipulation of the value, we can regenerate the password that used that same seed value in the past.

The chosen password generation parameters (A-Z, a-z, 0-9, hexadecimal, special characters, minimal number of digits, password length) also play a role in generating the password, so if we want to compute a previously generated password, we would need to have those parameters match the original. If these are unknown, then we can simply precompute all passwords for a given date range using each permutation of parameter settings.

As for the manipulation of system time before it is passed to _srand, after each "Generate New" button press, the code will subtract 0xe3a78 (decimal 932472) from a 32-bit int counter. This counter value is subtracted from the system time before calling _srand() on the next "Generate New" button press, which causes the time value to be incremented by a fixed amount - around 10.8 days (subtracting a negative number causes an addition, so time will "increase"). We believe this was added to prevent the same password from being generated multiple times if the user presses the "Generate New" button multiple times within one second. However, a side effect of this behavior is that even without any attack against the RoboForm code, it may be possible to generate a password that could be regenerated again in the future if the "Generate New" button is pressed at the correct time.

After identifying the area of interest within the roboform.dll using static analysis, we moved to dynamic analysis using x64dbg. This let us run the actual RoboForm password generator GUI in real time and debug the application (set breakpoints, modify data) while it was running. This is where we can prove that controlling the value fed into _srand will let us control the output password and, thus, regenerate any password that has previously been generated. We set a breakpoint at address 0x790853f2 which is after system time has been read and manipulated but before _srand is called (this memory address will vary depending on where the host computer loads the DLL into its system memory). At that point, the time value is stored in the register eax. By changing the value in the eax register to that of one earlier in time and then continuing execution of the application, we can regenerate any password that had been generated in the past. As an example, an eax value of 65b5c4b5 (decimal 1706411189) results in a password of o3p&hZWaH4Fn$T@3QZBm using parameters of 20 character password length, A-Z, a-z, 0,9, minimal number of digits 1, and special characters of !@#$%^&*

After proving that we could change the time value manually using the debugger while running the RoboForm GUI, we then created our own wrapper code that would hook into the password generator function directly within RoboForm's DLL (bypassing the GUI altogether) to generate any desired date range of passwords en masse. This wrapper contains a loop that configures the system time, passes the required configuration/parameters to the hooked password generator function within the DLL, and displays the resulting generated passwords.
