AEG: Automatic Exploit Generation

AEG is the first end to end system that automatically finds vulnerabilities and generate exploits for them.

Before this paper for control flow exploit generation human think very hard about whether a bug can be exploited or not, but authors develop techniques for automatic exploit generation on real programs in this paper.

In general way AEG searches for bugs at the source code, detects the error and gives us the exploit string. Lastly AEG runs the program with the generated exploit and verifies that it works.

The challenges in practical AEG:

1) Source code analysis is insufficient, binary level analysis is unscalable so authors combine them.
2) It is important that which exploitable path should be check firstly among an infinite number of possible paths, so authors employ symbolic execution to find out which path is exploitable and also use path prioritization technique to prioritize which paths should explore first.

Overview of the components of AEG:

AEG has six components:

1) PRE-PROCESS:
   The source program is compiled down to a binary and a LLVM byte code. These are AEG inputs.
2) SRC-ANALYSIS:
   AEG searching for the largest statically allocated buffers of the target program.
3) BUG-FIND:

This component receives LLVM bytecode and a safety property Ø as inputs, the output is $<\pi_{bug}, V>$ for each detected vulnerability. π_{bug} contains the path predicate and V contains source-level information about the detected vulnerability.

4) DBA (dynamic binary analysis):
   DBA receives binary and $<\pi_{bug}, V>$ as inputs then find a set of runtime information R.
5) EXPLOIT-GEN:
   This component receives π_{bug} and R then generates control flow of an exploit. AEG produces two type of exploits: return-to-stack and return-to-libc, which both of them change the return address.

6) VERIFY:
   This component receives binary and exploit formula, if there is a satisfying answer it will return ε and AEG runs the binary with ε as an input to checks if the adversarial goal is satisfied or not, otherwise if there weren't satisfying answer, AEG fails to generate the exploit.