Software Vulnerabilities: Exploitation and Mitigation

Artem Kaliahin
artem.kaliahin.001@student.uni.lu

March 2019

Question 1.1 What is BROP? What is the difference between ROP?

BROP is a blind return oriented programming. This technique remotely locates ROP gadgets. The main difference is that BROP attack makes it possible to write exploits without possessing the target's binary, when to perform ROP attack we need to know the location of ROP gadgets within the code. Unlike ROP attack, BROP attack can be also performed when ASLR is enabled.

Question 1.2 Does a BROP attack also work for binaries for which the attacker does not have access to (no source, no binary)? Explain.

Yes, BROP attack works for binaries for which the attacker doesn't have access to. It remotely finds enough gadgets to perform the write system call, after which the server's binary can be transferred over the network from memory to the attacker's socket. It allows attacker to have a copy of vulnerable binary and hack closed-binary services where we don't have access to target's binary code.

Question 1.3 Look at Figure 4. Is there a situation where the BROP approach does not work? Why?

Since in the BROP attack we use stack reading technique that allows us to read the stack by overwriting it byte-by-byte with possible guess values, until the correct one is found and the server does not crash. Therefore, BROP approach doesn't work if the server was compiled PIE-flag enabled AND rerandomizes its memory after crush since our stack reading technique doesn't work in these conditions.

Question 1.4 Does a BROP attack also work for binaries for which the attacker does not have access to (no source, no binary)? Explain.

Look question 1.2

Question 1.5 What are the requirements to perform a BROP attack? Explain.

First of all, we need the stack vulnerability itself and to have an understanding on how to exploit it. Moreover, BROP attack requires a server application that restarts and DOES NOT rerandomize memory after a crush (if compiled with PIE flag). We have to be able to crash server as many times as we need as well as get response from it if it's been crashed to perform the stack reading (server will not crash in case it's overwritten with the same value).

Question 1.6 Instead of randomly brute-forcing the 264 address space, an attacker can leak the return address. How does an attacker leak the return address? Explain.

Return address have to be leaked within 1st phase of BROP attack: stack reading. The idea is to overflow every single byte one by one until server won't get crashed (it shows us that the value is correct). This allows us to leak all 8 canary bytes values. In most cases, saved return address is located after saved frame pointer, which is located after canary which protects saved registers.

Question 1.7 Why are stop gadgets? What are they useful for? Explain.

Stop gadgets are gadgets that stop program from crashing or indicate to the attacker that one has been executed. It helps attacker to identify useful gadgets. After finding useful gadget, the application will most likely crash because of its attempt to return to invalid address. So in order to "catch" useful gadgets, we need to stop ROP chain execution. Each time program will find a gadget that doesn't cause crash of the application, stop gadget will execute and indicate that useful gadget has been found.

Question 1.8 To classify gadgets, different stack layouts (the data the attacker puts on the stack) are used? What are the different stack layouts and how can they be used to identify categories of gadgets? Explain.

There is infinite amount of stack layouts attacker can use to identify gadgets by varying the position of the following values on the stack:

1. probe (address being scanned);
2. stop (address of a stop gadget);
3. trap (leads to non-executable memory => crash).

By changing the stack layout, we can mark gadgets that do or don't pop words from the stack. For instance, the following layout - probe, trap, stop, traps -

we scan "probe" address of memory, then it will find gadgets that pop 1 stack word (if "trap" will be executed rather then popped, the program will crash and the opposite - if "trap" will be popped from the stack, program will not crash because it jumps to our stop gadget).

Question 1.9 To execute a function, the attacker has to find the PLT. What is the PLT? How can an attacker find this table? Explain.

PLT is a Procedure Linkage Table that is needed for non-PIC (position-independent code) that is dynamically linked. Since PLT is often pretty big, has a lot of entries and most of them don't cause a crash regardless the given arguments, attacker can scan program's address space and if a couple of addresses 16 bytes apart do not cause a crash (as well as these addresses with offset of 6 bytes to verify that this is PLT), he can be pretty sure that PLT is found.

Question 1.10 How long, did it take to exploit a vulnerability with a BROP attack in yaSSL + MySQL? in nginx?

For yaSSL + MySQL exploiting the vulnerability took 20 minutes (since on every crash script restarts the server). As for nginx attack, it took only one minute because of non-time-based stop gadget.

Question 1.11 The return address might not be recovered if a load balancer is used (we assume here attack on a remote host through the network). Explain why.

With load balancing architecture network traffic distributes through a group of servers, which prevents attacker from contacting back-end servers directly. But the attack itself relies on attacking THE SAME machine and process. Therefore, such architecture have security benefits by hiding the structure of the internal network and preventing attacks on the kernel's network stack.

Question 1.12 The attack might not work if all workers are stuck in a loop. Explain why.

In applications that are configured the way that they don't fork per connection or don't have multi-threaded workers, there can be a situation, when all workers are running infinite-loop-based stop gadgets, which makes all workers being stuck in the loop. Therefore, it is impossible to continue the attack since all workers become unresponsive.
The solution is not to use infinite-loop-based stop gadgets, but gadgets that return to a higher stack frame.