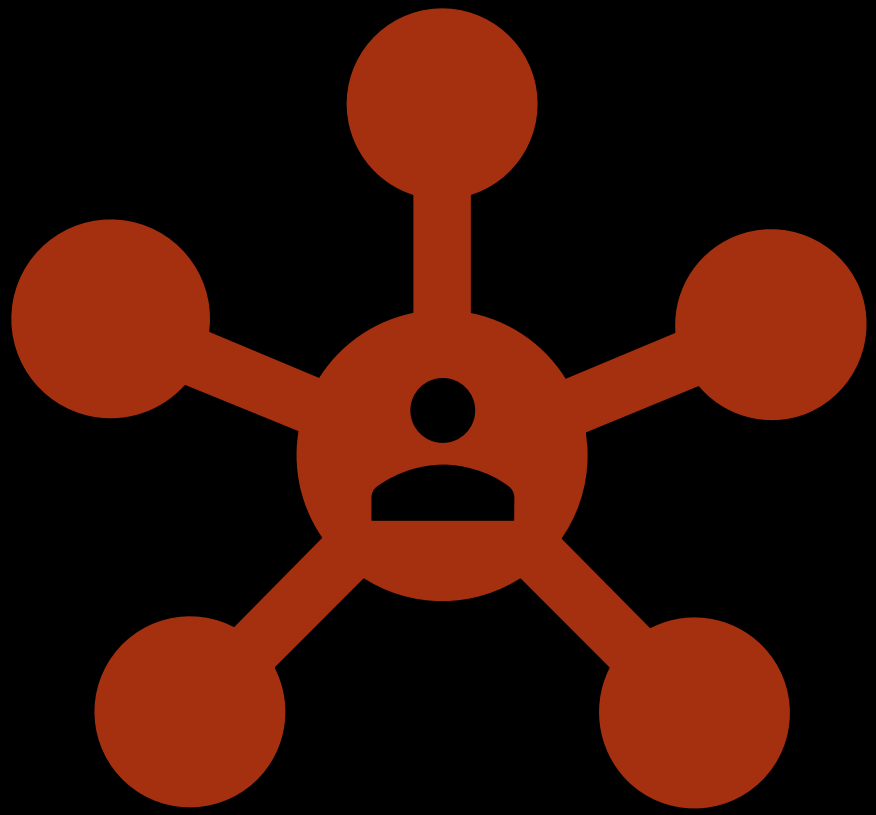
Evading Exploit Development Defense With ROP

ABOUT ME:

- MANISH SHARMA
- APPLICATION SECURITY ENGINEER (3 YRS)
- OPEN FINANCIAL TECHNOLOGIES
- GITHUB/SH377COD3
- @SH377COD3
- LINKEDIN/IN/SH377COD3

ME MYSELF AND I.

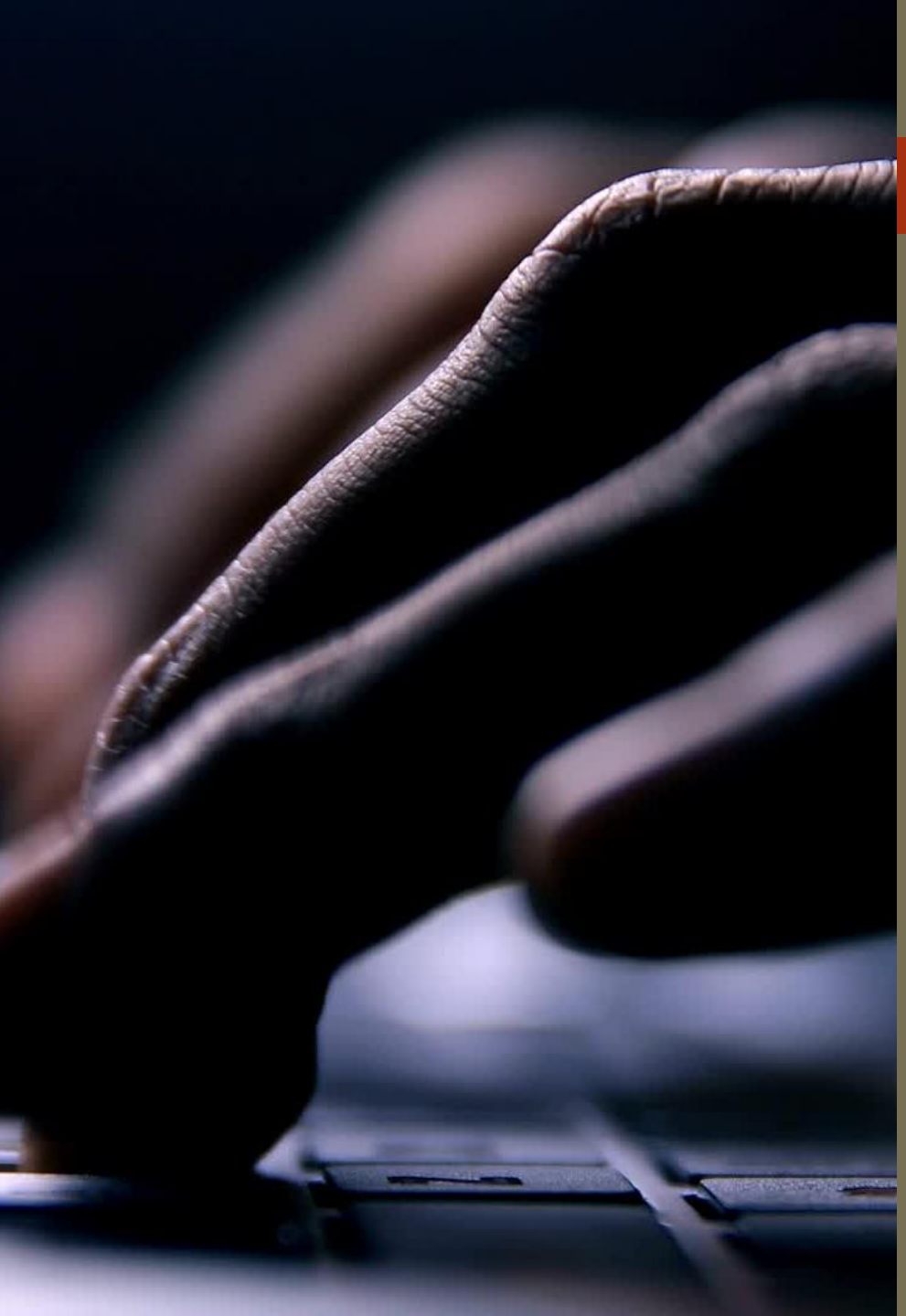


Disclaimer

- This entire talk is more at personal level and doesn't contain or relate to any of my former or current professional associations.
- All content has been sourced from publicly available sources like the internet. The presenter does not infringe or claim rights over any content, images and any other work being presented here, and all rights belong to respective owners only.

AGENDA

- What is Exploit Development
- Exploit Dev Defenses ?!
- ROP! Intro?
- Windows vs Linux
- Reference
- Q & A

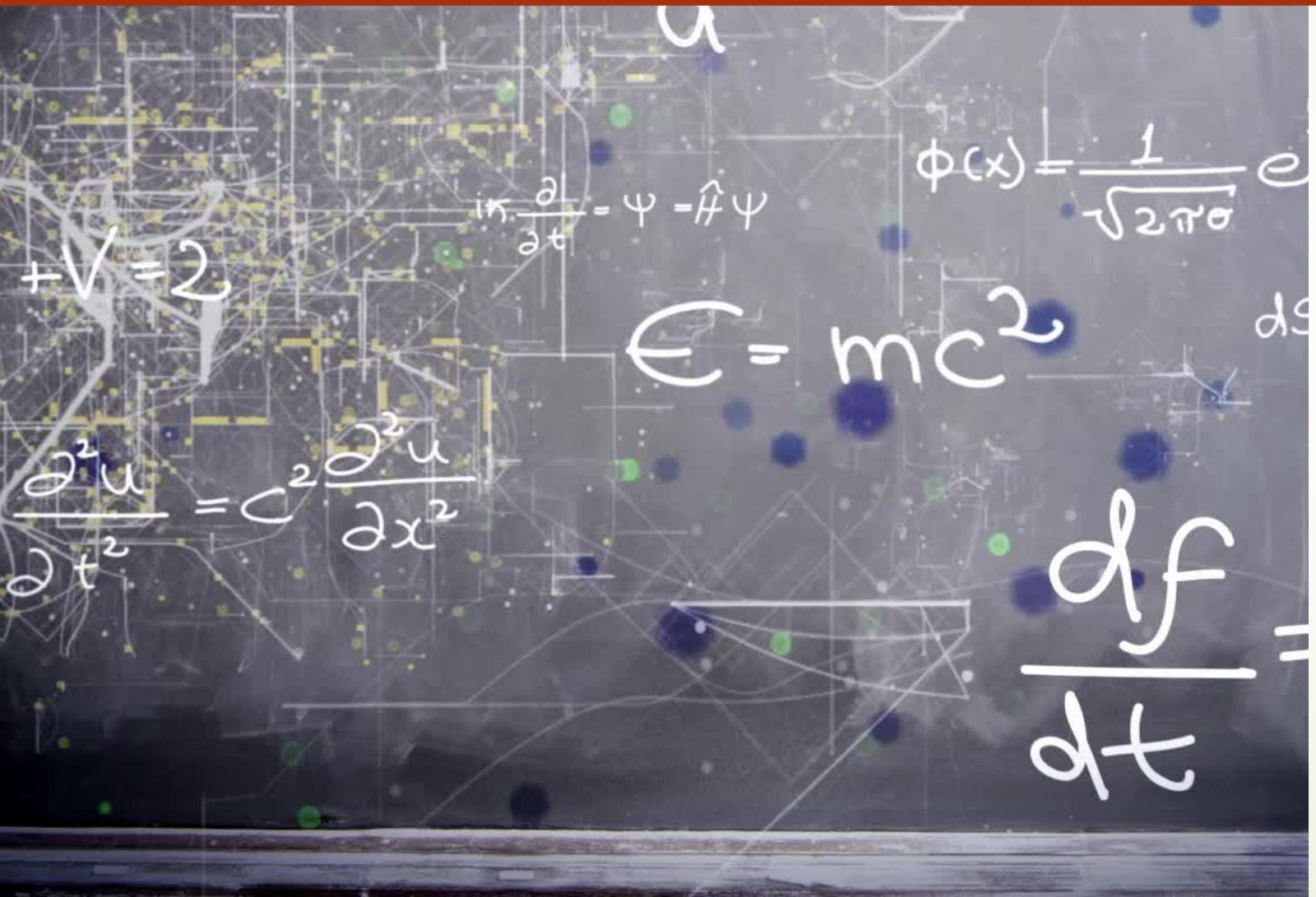


What is Exploit Development

ChatGPT says:

- Exploit development is the process of finding, creating, and developing software or code that takes advantage of a vulnerability in a computer system, network, or application to cause unintended or unauthorized behavior.
- The goal of exploit development is often to gain control of a system, steal sensitive information, or cause damage.

What is Exploit Development(cont.)



- Fuzzing
- Finding offset
- Overwriting EIP
- Finding Bad characters
- Finding Right Module
- Generating shellcode
- Exploit !!!

Exploit Dev Defense ?!

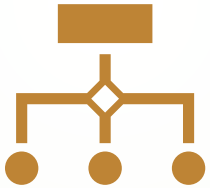
- Stack Canaries
- Format String protection
- Address Space Layout Randomization (ASLR)
- Data Execution Prevention (DEP)
- Read-Only Data protection
- SafeSEH

Exploit Dev Defense ?! (contd.)

- There is only ASLR, you could brute-force the shellcode address.
- Only NX, you could return to libc, as it is always at the same address.
- What if … there is ASLR + NX? -- Can't brute-force now !
- Can't return to the system, as it will always be at a different address.
- Now How to Achieve Exploitation ?!

ROP! Intro?

Return-Oriented Programming is successor of return-to-libc attack technique.



In return-oriented programming, you can chain multiple functions to form a ROP chain.

Gadgets? These are nothing but sequence of code residing in executable memory followed by return instruction.

ROP! Intro? (contd.)

Gadgets and Return ? ? ?

- 1122aa33 holds the real, intended instruction
- Let's offset it 1 byte and now it points to 1122aa34
- Just 1 byte off and completely different instructions followed by a return!
- This is how gadgets are built !!!

ROP! Intro?(contd.)

Abuse code that is:

- Already within the process address space
- Not randomized (remember that ASLR randomizes certain sections, not everything

There could be another function instead of gadget().

The only thing that should be done is that the stack should be prepared for another function.

ROP !!!

1. Explore and achieve Overflow vulnerability
2. Overwrite return address program with a ROP gadget…
3. ROP gadget pop a value from stack and store it in register
4. Now find out another ROP gadget for a specific function
5. Chain gadgets to pop value into a REGISTER
6. Execute second gadget to perform another specific operation
7. BOOM !! ROP chain is executed calling vulnerable function

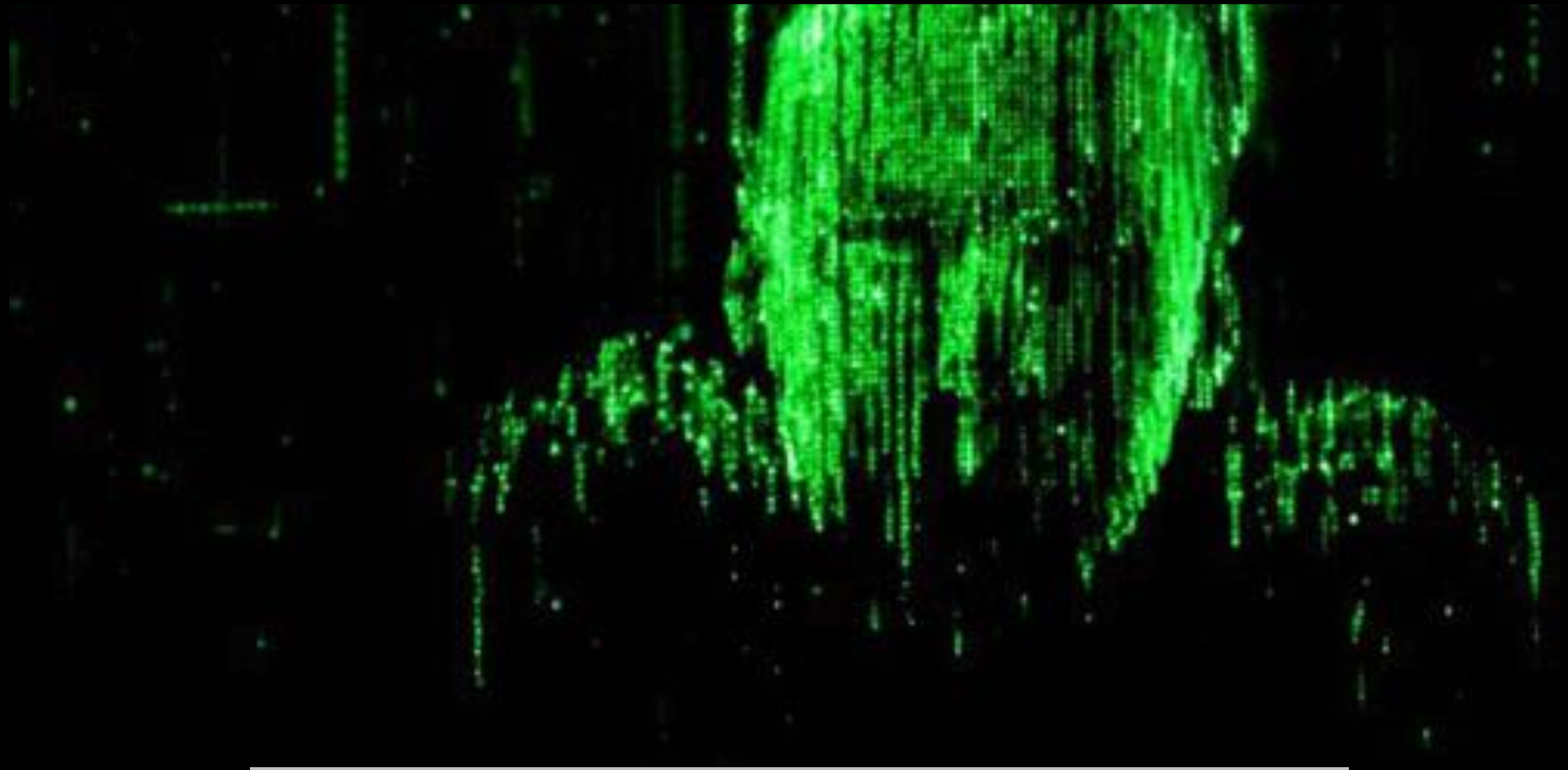
ROP !!! (contd.)

- ✓ Hovav Shacham and Stephen Checkoway released a paper on ROP without returns:

  https://hovav.net/ucsd/dist/noret.pdf

- ✓ The idea is to get around some protections that may search through code looking for instruction streams with frequent returns.
- ✓ Another defense attempts to look for violations of the LIFO nature of the stack.

Windows vs Linux



Let me Explain more about this with Demo’s

REFERENCE

- https://hovav.net/ucsd/dist/noret.pdf
- https://github.com/JonathanSalwan/ROPgadget
- https://github.com/sashs/Ropper
- https://github.com/ctfs/write-ups-2013/
- https://reboare.github.io/bof/linux-stack-bof-3.html
- https://devel0pment.de/?p=366
- https://www.scriptjunkie.us/2011/03/finding-non-aslr-or-dep-modules/
- https://docs.microsoft.com/en-us/Windows/win32/api/memoryapi/nf-memoryapi-virtualprotect

THANK YOU