Mecanismos anti-exploits em sistemas Linux

RONER RODRIGUES

DISCENTE DE ENG. DE COMPUTAÇÃO

DELPHI + SQL DEV @COBRAZIL.COM.BR

Tópicos

- O que são exploits ?
  - Classificação e funcionamento
  - Exploits famosos
- Vulnerabilidades e técnicas de exploiting comuns
  - Buffer overflow
  - ROP chain
  - ret-to-lib / ret-to-dll
- Upgrades de segurança no kernel linux
  - W^X
  - ASLR / KASLR
  - KARL
- Conclusão

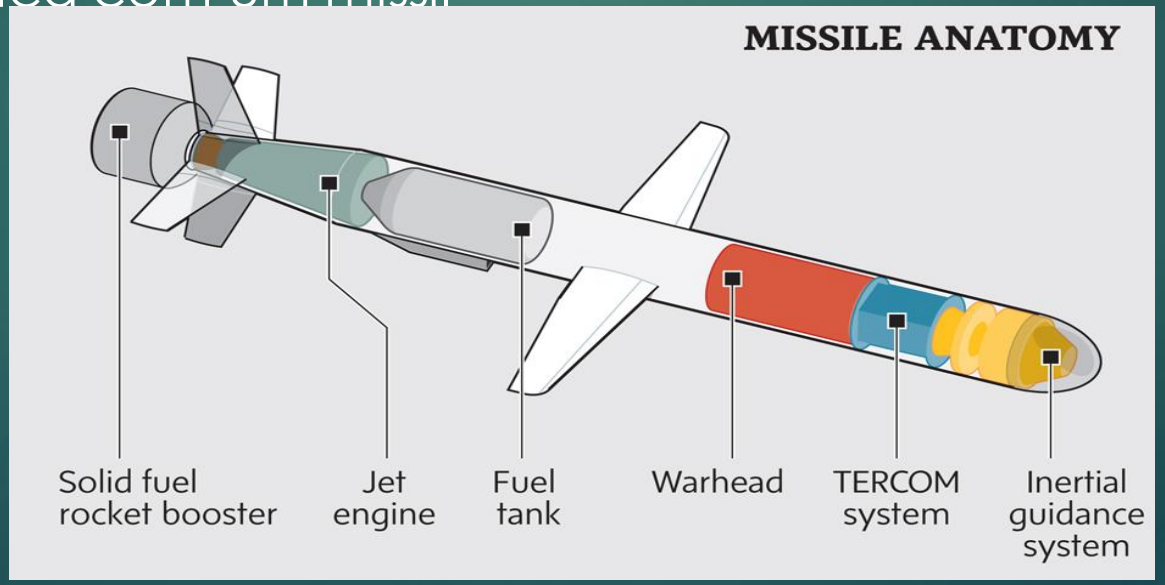
O que são exploits ?

- Exploit é o pedaço de código responsável pela exploitação - ação de explorar uma vulnerabilidade
  - stack / heap overflow
  - ret-to-lib / ret-to-dll
  - ROP - Programação Orientada a Retorno
  - Falhas web, etc.
- Aplicações diversas
  - Escalação de privilégios
  - Negação de serviço (DoS)
  - Cracking de jogos online (dll injection)

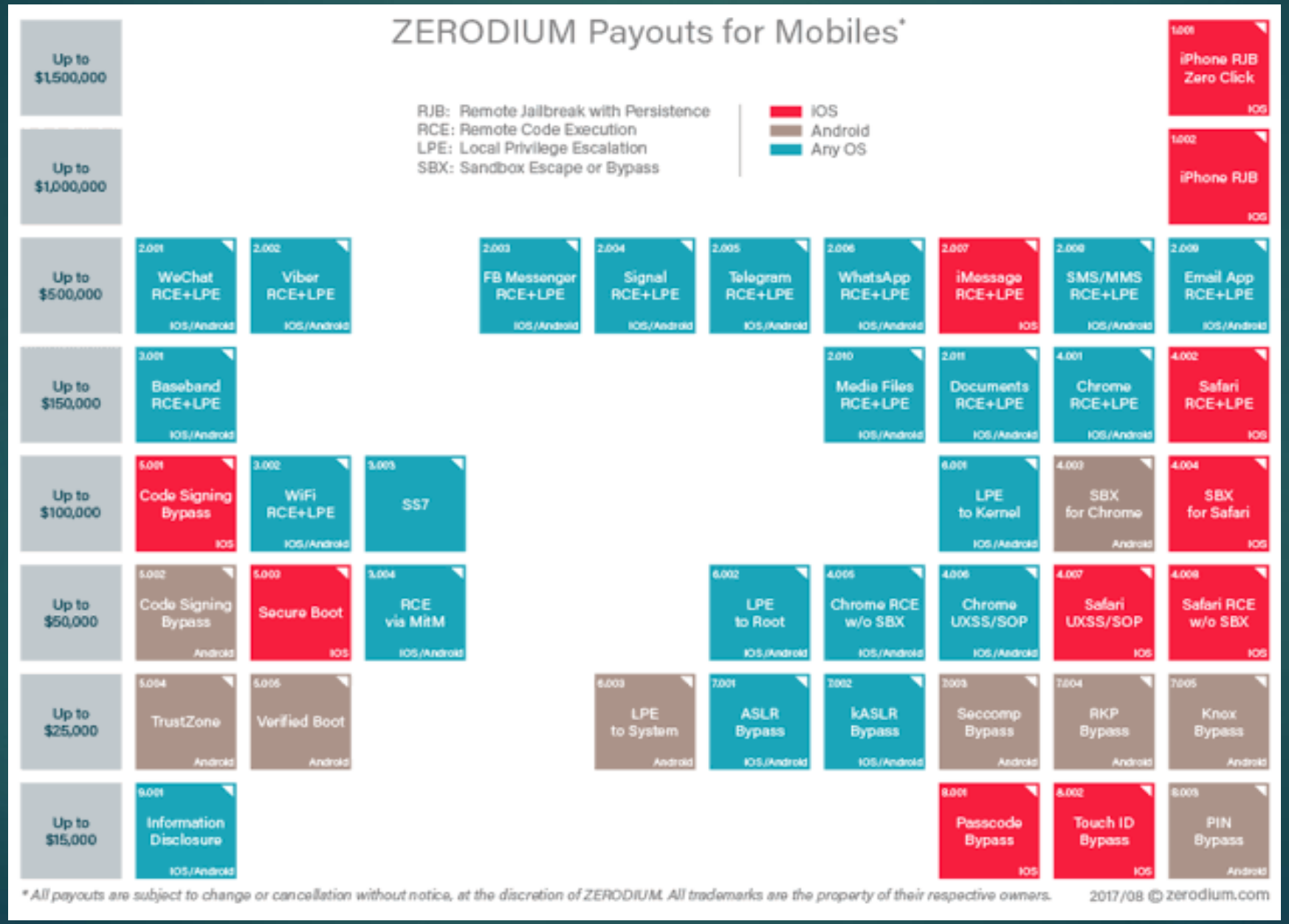
- O maior problema sempre vai ser o dia-zero - não dá para evitar!!

Funcionamento

- Para alguns autores, exploits consistem de 2 estágios
  - **(1)** Exploit – Aproveita-se de uma vulnerabilidade para (geralmente) desviar o IP para uma área de memória contendo código injetado pelo atacante (payload).
  - **(2)** Payload / Shellcode - Código que será executado com fins diversos: escalação de privilégio, negação de serviço, conexão reversa, etc
- Analogia clássica com um míssil



Valores para exploits 0-day



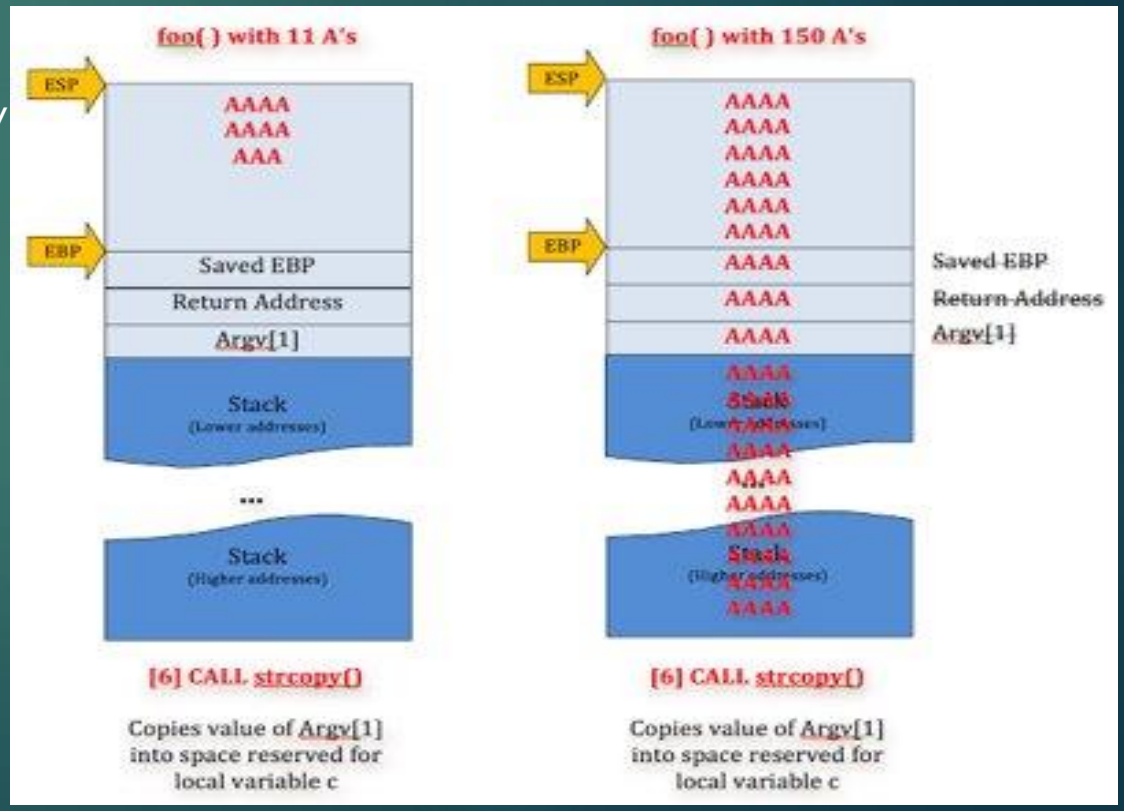
Exploits famosos

- CVE-2010-2568 – Uma das 4 falhas 0-day utilizadas no cyberataque contra o Irã no programa STUXNET. Consiste de código malicioso inserido em arquivos .LNK em drivers USB externos, que são lidos pelo Windows para exibição de ícones de atalho.
- CVE-2017-5754 : Meltdown – Vazamento de informação do kernel através do recurso de execução especulativa, o que abrange quase que 20 anos de processadores fabricados.
- CVE-2017-5753 : Spectre – Permite enganar programas 'livres de erro' e que seguem as 'melhores práticas'. Mais difícil de exploitar do que o Meltdown, porém mais difícil de mitigar também.
- KRACK – Vulnerabilidade no handshake WPA2 em alguns dispositivos wifi que permite vazamento de informação. Estima-se que pelo menos 41% dos dispositivos Android são vulneráveis, assim como produtos Apple, Windows, OpenBSD e outros.
- CVE-2017-0144 : EternalBlue – Cyberarma desenvolvida pela NSA para auxiliar em suas operações. Se aproveita de vulnerabilidade no protocolo SMBv1 que permite execução remota de código. Utilizada pelo WannaCry e ransonware semelhantes.

Buffer overflow

- Buffer overflow é a designação para uma família de vulnerabilidades que surgem a partir da falta de verificação devida de entrada do usuário:
  - stack / heap / .data / .bss overflow / Format strings
- Práticas de desenvolvimento seguro incluem checagem e fix de pontos de entrada vulneráveis: bounds-checking em runtime + proteções do compilador
- Aplicações C/C++ são mais vulneráveis
- Limitação: tamanho e permissões do buffer alv

Insecure Functions	Safe Functions
strcpy()	strlcpy()*, strcpy_s()*
strcat()	strlcat()*, strlcat_s()*
printf() /	snprintf()*, sprintf_s()*
gets	fgets()
*Functions do not fall under C Standard Libraries.	

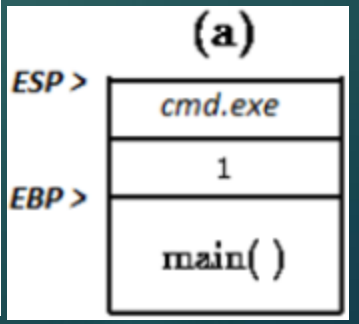


Return to Library / DLL

- ret-to-lib / ret-to-dll : se não podemos executar código inserido na pilha, sem dúvidas podemos desviar o EIP para bibliotecas do sistema onde devemos ter essa permissão
- Limitação: pelo seu funcionamento, depende da convenção de chamadas comuns da arquitetura intel x32, onde parâmetros são passados na pilha

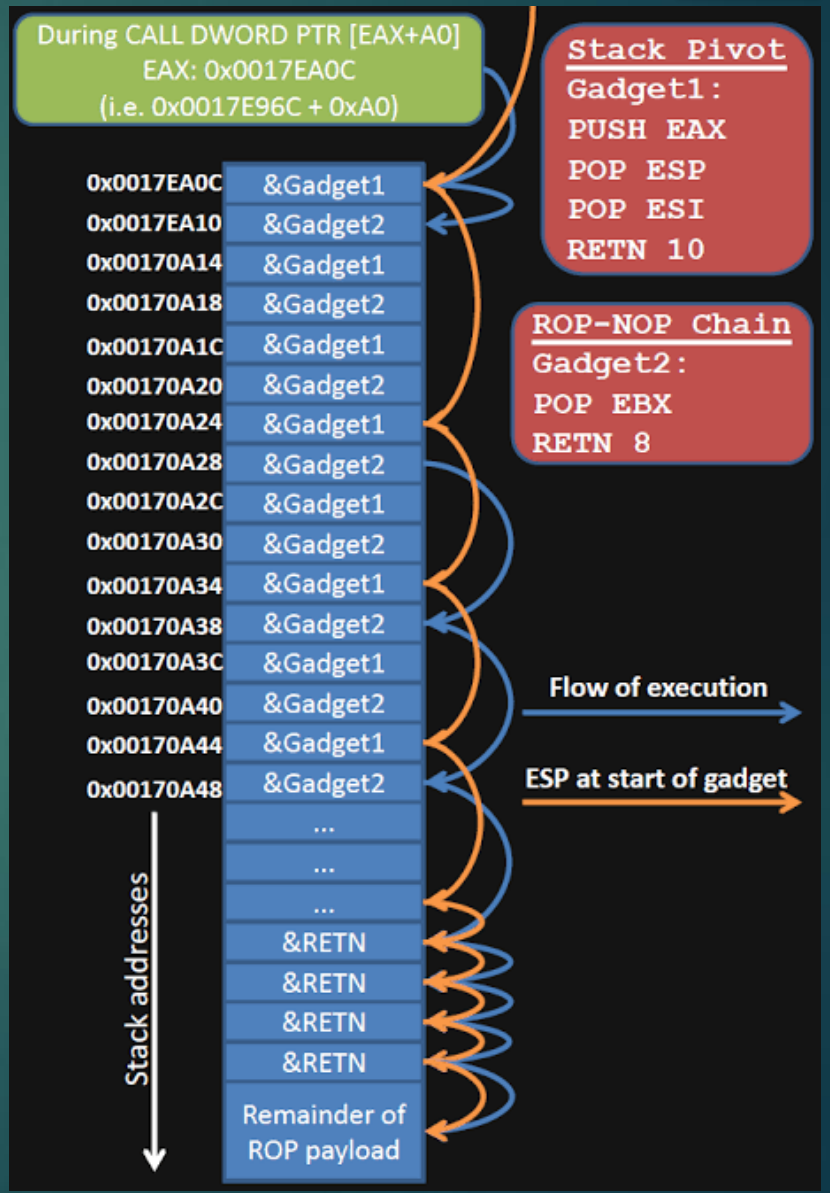
Função *main()*

```
                push    ebp
                mov     ebp,esp
                sub     esp,48h
                push    ebx
                push    esi
                push    edi
                mov     eax,dword ptr [Summon!`string' (00415b30)]
                mov     dword ptr [ebp-8],eax
                mov     ecx,dword ptr [Summon!`string'+0x4 (00415b34)]
                mov     dword ptr [ebp-4],ecx
                push    1
                lea     eax,[ebp-8]
                push    eax
EIP > (a) —     call    dword ptr [Summon!_imp__WinExec (00418000)]
                xor     eax,eax
                pop     edi
                pop     esi
                pop     ebx
                mov     esp,ebp
                pop     ebp
                ret
```



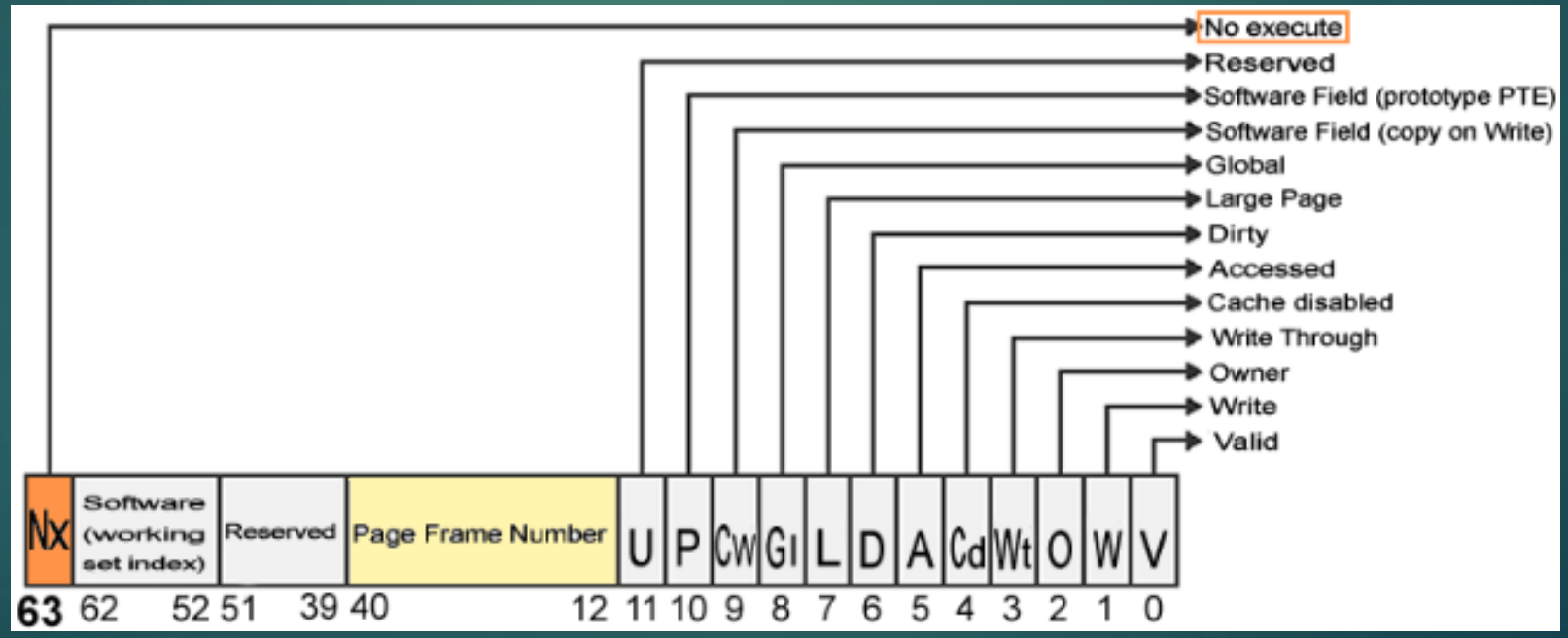
ROP chain

- ROP - Programação Orientada a Retorno
- Se não podemos executar código malicioso inserido na pilha:
  - Encontrar instruções (gadgets) do próprio programa-alvo para montar nosso shellcode em tempo de execução
- Limitação: disponibilidade de gadgets
  - Compilador performa remoção de returns indesejados + re-ordenação de registradores
  - Não precisa reduzir a 0 gadgets: apenas o suficiente para inviabilizar o ROP chain
  - Ex: OpenBSD ativamente reduz o número e variedade de gadgets disponíveis



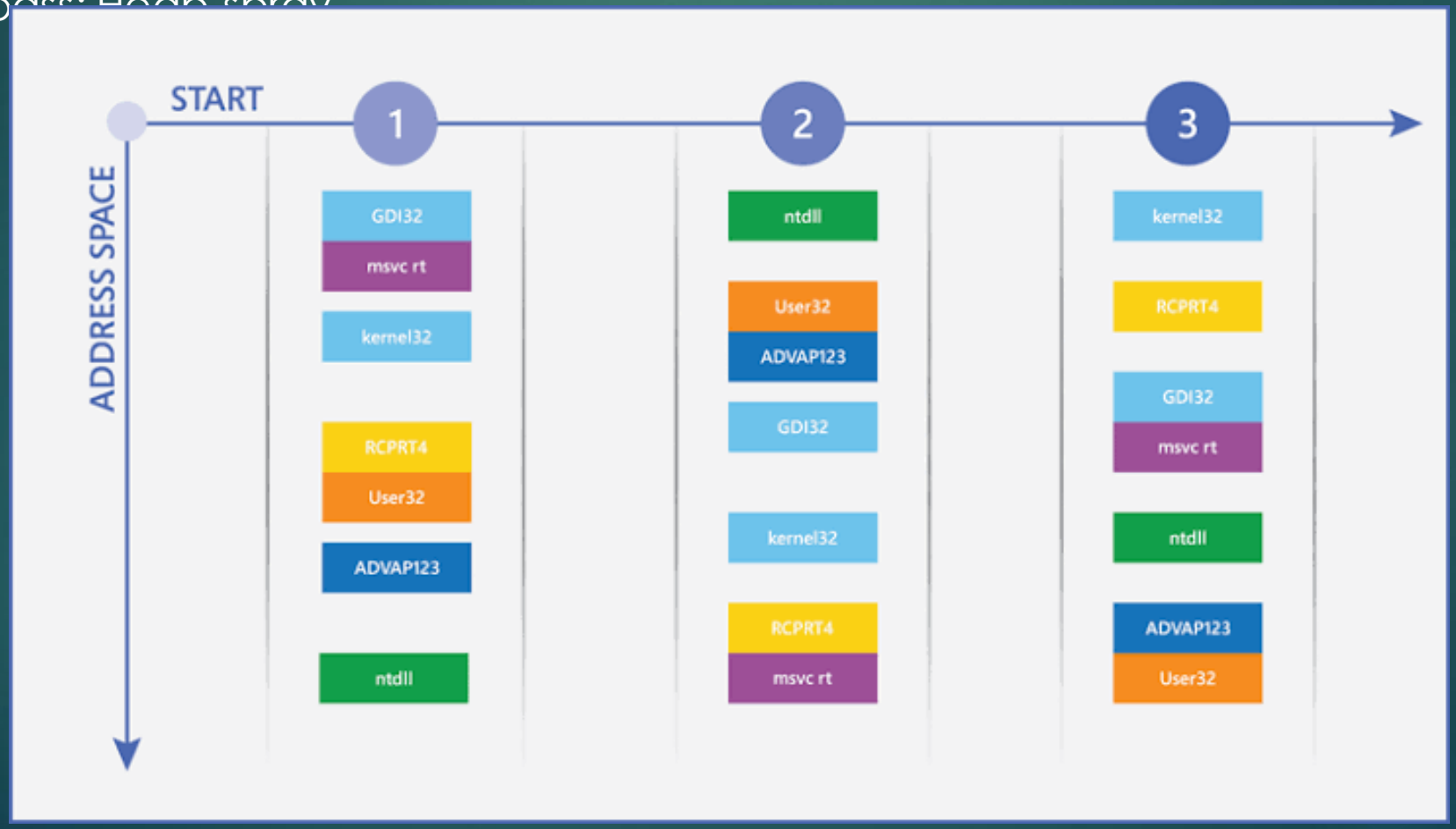
Writable xor eXecutable (W^X)

- A partir do kernel 2.6.8, páginas do kernel podem ser marcadas com o flag **NX** (x64)
  - Funciona mesmo sem a disponibilidade do bit NX no processador (PAE)
  - Já habilitada por padrão em algumas distros como Hardened Gentoo, Trusted Debian, OpenBSD e outras
  - Bypass: ret-to-lib, ROP chain



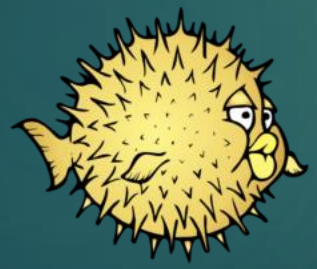
ASLR / KASLR

- Implementado por padrão no kernel desde 2006 (kernel >= 2.6)
- Posição de segmentos e libraries randomizados no momento da inicialização do programa
- Bypass: Heap spray



Kernel Address Randomized Link - KARL

- Um novo kernel (do ponto vista binário) a cada novo boot
- Frustra consideravelmente o processo de exploitação
- Ao invés de randomizar o endereço base para todo o código do kernel a cada boot, randomiza a ordem do linkagem de arquivos objetos por seus binários:
  - Divide o assembly runtime (*.S) de locore.o, linkando aos outros .o do kernel de forma random
  - Com isso, o vazamento de informação do endereço de uma única função do kernel não vai revelar a localização de todas as outras funções
  - São poucas mitigações contra ataques BROP, mas que podem ser resumidas em "nunca reutilize espaços de enderaçamento"



OpenBSD

Muito Obrigado !