Disciplina: Segurança Cibernética

LazyFragmentation Heap (WCTF 2019)

Grupo
Arthur Hugo Barros Gaia
Felipe Ivo da Silva
Nathalia Cristina Santos
Thiago Zucarelli Crestani
Wilson de Camargo Vieira

SUMÁRIO

1. Introdução (Thiago)
2. Ambiente de Testes (Thiago)
3. Scripts (Thiago)
4. Execução e Evidência (conectividade) (Nathalia)
5. Análise de *dumps* (o que procurar) (Nathalia)
6. Estratégias quando não há *leak* (Felipe)
7. Plano para construir *exploit* final (Felipe)
8. Resultados parciais e próximos passos (Welison)
9. Conclusão (Welison)

Introdução

- Reproduzir o binário alvo e torná-lo acessível na rede.
- Automatizar a sequência do writeup (warmup LFH → A,B,C → free(B) → grooming → LazyFileHandler → show(A)).
- Coletar dumps e detectar vazamentos (endereços, cabeçalhos PE).
- Construir exploit final (ROP + shellcode) a partir dos leaks.
- Documentar todo o processo para apresentação.

Ambiente de Testes

- **Host:** Debian 13 (KVM) - VM atacante: Kali (com Docker).
- **Container:** lazyfrag_pwn (Docker) com Python3 + pwntools.
- **Alvo:** Windows 10 (KVM) rodando AppJailLauncher.exe / LazyFragmentationHeap.
- **Rede:** libvirt NAT default (endereço alvo observado: 192.168.122.93).
- **Porta do serviço:** TCP 6677.

Scripts

- **exp_auto.py:** helper interativo; warmup + demo grooming.
- **exp_bruteforce2.py:** brute-force agressivo de combinações.
- **exp_final.py:** reproduz exatamente o passo-a-passo da resolução; salva dumps e summaries (https://github.com/scwuaptx/LazyFragmentationHeap/blob/master/LazyFragmentationHeap_slide.pdf)
- **exp_full.py:** template de exploit (preenchimento automático após leak).

Scripts (saídas)

- **Local de saída padrão (no container):** /tmp/exp_final/run_<TIMESTAMP>/
- Arquivos gerados por tentativa:
  - **trial_<N>_show.bin:** saída do show(A) (blob salvo).
  - **trial_<N>_lazyhandler.bin:** saída do LazyFileHandler.
  - **trial_<N>_summary.json:** JSON com campos addresses (lista de 0x...) e strings (lista de strings detectadas).
  - **master_summary.json:** resumo da execução.

4. Execução e Evidência

COMPLEX FILE SERVICE

Desafio do DAMCTF de 2025

5. Análise de dumps

COMPLEX FILE SERVICE

Desafio do DAMCTF de 2025

Estratégias quando não há *leak*

- Aumentar *warmup* (mais alocações do mesmo tamanho) e variar tamanhos de chunk.
- Repetir o fluxo muitas vezes (incrementar repeat) e rodar por longos períodos.
- Testar parâmetros alternativos: base=0/1, editar antes de free, variar extra/grooming.
- Variação aleatória controlada: variar offsets/timings entre execuções.
- Rodar brute-force agressivo (scripts automatizados) para ampliar cobertura de estados.
- Monitorar artefatos (trial_*.bin, summary.json) e automatizar análise (xxd/strings/assinaturas).

Plano para construir exploit final

- Obter leak confiável (pré-requisito).
- Calcular *base* do módulo: usar o leak identificado e subtrair o offset conhecido para inferir a base.
- Mapear regiões/endereços úteis do processo (apenas para análise offline).
- Localizar gadgets e estruturar *conceptual* ROP chain (sequência lógica, não código).
- Escolher payload apropriado (ex.: POC inofensivo / demonstração controlada) e preparar ambiente de testes.
- Testar incrementalmente em VM isolada; validar apenas efeitos controlados (logs, spawn de processo de teste).

Muito obrigado!