TLP: CLEAR

Pesquisa de Cibersegurança

Cyber Threat Actor

LockBit 5.0: Análise Técnica da Nova Geração do Ransomware Mais Prolífico do Mundo

heimdall
security research

Acesse nossa comunidade no WhatsApp, clicando na imagem abaixo!

heimdall
security research
A DIVISION OF ISH

Acesse as análises produzidas pela ISH Tecnologia sobre Táticas, Técnicas e Procedimentos (TTPs) de Threat Actors, malwares emergentes, vulnerabilidades críticas e outros temas relevantes em cibersegurança. Clique na imagem abaixo para conferir nosso blog.

ISH

ALERTA HEIMDALL! HTTP2 RAPID RESET_IMPACTOS E DETECÇÃO DA CVE-2023-44487

Falhas de negação de serviço (DoS) não são apenas interrupções técnicas. Elas representam riscos reais à continuidade do negócio, à confiança dos clientes e à reputação da marca. Nisto temos a vulnerabilidade CVE-2023-44487, conhecida como HTTP/2 Rapid Reset.

BAIXAR

ISH

ALERTA HEIMDALL! BABUK2 EM 2025: RETORNO LEGÍTIMO OU COPYCAT ESTRATÉGICO

O cenário de ransomware manteve-se ativo em 2024 e se estendeu para este ano de 2025, com diversos grupos realizando ataques a uma ampla gama de organizações, setores e com surgimento de novos grupos. Nesse contexto, temos o ransomware Babuk2.

BAIXAR

ISH

ALERTA HEIMDALL! A ANATOMIA DO RANSOMWARE AKIRA E SUA EXPANSÃO MULTIPLATAFORMA

O cenário de ransomware manteve-se ativo em 2024 e se estendeu para este ano de 2025, com diversos grupos realizando ataques a uma ampla gama de organizações, setores e ganhando bastante popularidade. Nesse contexto, temos o ransomware Akira.

BAIXAR

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

PRINCIPAIS ATUALIZAÇÕES DO LOCKBIT5.0

Lockbit é um dos poucos grupos de Ransomware, que de fato avançam nas implementações de suas capacidades, sempre que implementam uma nova versão do seu Ransomware.

Esta nova versão par Windows não é diferente, pois implementa novidades, desde o menu de ajuda (-h) estilizado, até o processo de *unpacking* e a reconstrução de sua Tabela de Importações. Esta nova versão, é de fato a versão mais avançada do Lockbit, pois, nela é implementada técnicas avançadas de Anti Análise e Evasão de Produtos de Segurança, com o objetivo de aumentar o nível de complexidade para os pesquisadores, e dificultar a sua detecção e resposta por meio de produtos de segurança. A versão do Lockbit5.0 para Linux e ESXi não encontram-se em estado *packed*, porém, implementam a mesma tática de Anti-Análise implementada pela versão para Windows. Por este motivo, e pelo fato dos principais alvos serem sistemas Windows, esta análise será focada no Lockbit5.0 para Windows.

Abaixo, é possível observar a atualização visual implementada no Lockbit5.0, um menu estilizado, deixando o operador ciente que ele está utilizando a versão 1.01 do Lockbit5.0.

```
PS C:\Users\WDAGUtilityAccount\Desktop> .\win64.exe -h

 LOCKBIT5.0  ChuongDoung Locker v1.01  Windows x64

USAGE
chuongdong64.exe [options]
  * Command line length is limited to 500 characters.

BASIC OPTIONS
-h                      Show this help
-p <dirs>               Semicolon-separated list of directories to encrypt
-b <dirs>               Semicolon-separated list of directories to bypass

OPERATION MODES
-i                      Invisible mode (don't change extensions, no notes, don't change modification date)
-v                      Run in verbose visible mode with status bar in console
                        * Not available when using -p
-d                      Run in visible mode with debug output

NOTES SETTINGS
-n <0/1/2>              Notes storage mode (0: none, 1: everywhere, 2: C:\ only)
                        * This option is ignored when using -i (invisible mode)

ENCRYPTION SETTINGS
-m <mode>               Encryption mode (all/local/net)
-f                      Fast encryption mode
-w                      Enable wipe free space after encryption

FILTERING
-k                      Don't delete .exe
-nomutex                Allow multiple instances
-t <seconds>            Set timeout before starting encryption

EXAMPLES

chuongdong64.exe
  Encrypt entire system with default settings

chuongdong64.exe -p "C:\Users;X:\remote"
  Encrypt C:\Users and X:\remote directories

chuongdong64.exe -m local -k
```

Figura 1 - Menu de Ajuda do Lockbit5.0

A equipe de pesquisa da ISH Tecnologia vem analisando de maneira profunda o Lockbit e suas versões, desde o Lockbit3.0, com o objetivo de compreender a ameaça a longo prazo, e identificar os avanços de capacidades implementadas a cada nova versão.

Com esta análise, é possível afirmar que a versão 4.0 do Lockbit, possivelmente era um teste operacional de capacidades a serem de fato implementadas na versão 5.0. Abaixo, segue as principais características que se encontram em cada versão.

Features	Lockbit3.0	Lockbit4.0	Lockbit5.0
Algoritmo Customizado de Hashing	Sim	Sim	Sim
Técnica de Escalação de Privilégios	Sim	Não	Não
Evasão de Defesa via ETW Patching	Não	Sim	Sim
Evasão de EDR via API Unhooking	Não	Não	Sim
Enumeração de Rede	Não	Sim	Não
Implementação de Algoritmo RC4	Não	Sim	Não
Coleta de Credenciais	Sim	Não	Não
Métodos de Persistência	Sim	Não	Não
Configuração do Safe-Boot Mode	Sim	Não	Não
Identificação da Região do Dispositivo Infectado	Sim	Sim	Sim

Tabela 1 - Presença de Técnicas em Cada Versão

Ao realizar uma análise comparativa destas principais implementações, das versões 3.0 a 5.0, podemos notar que o grupo Lockbit passou a desenvolver um Ransomware mais objetivo e com o objetivo claro de evadir produtos de detecção, mantendo técnicas como o ETW Patching, e adicionando à versão 5.0 a capacidade de evadir produtos de segurança como *EDRs*, por meio da implementação da técnica de API Unhooking.

ENGENHARIA REVERSA DO LOCKBIT5.0

A amostra do Lockbit5.0 que é executada nos sistemas vítimas, encontra-se criptografado (*packed*), e durante a sua execução a amostra executa um processo customizado de descriptografia (*unpacking*), e executando o *payload* final em um processo remoto, por meio da técnica de Process Injection.

EXTRAÇÃO DO LOCKBIT5.0 DESCRIPTOGRAFADO

Durante o processo de *unpacking*, um novo processo do binário nativo de desfragmentação do Windows (defrag) é criado, e o Injector por sua vez, aloca e injeta o *Lockbit5.0 descriptografado* em um espaço na memória do processo remoto (defrag.exe), na qual será posteriormente executado dentro do contexto do processo remoto.

Na imagem abaixo, é possível observarmos que o Injector utiliza a API de Nativa NtCreateUserProcess, para criar o processo alvo (defrag.exe).

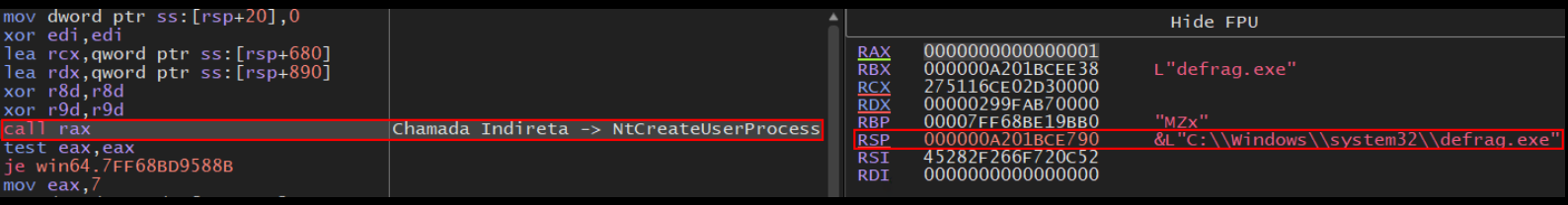

Figura 2 - Criação de Processo Alvo de Injeção

Na visualização de registradores do *debugger*, é possível analisarmos alguns valores armazenados em registradores, sendo um deles os Magic Bytes referente a um *DOS Header* de um PE (*Portable Executable*), o 'MZ', presente em um endereço em memória armazenado no registrador RBP. Abaixo é possível observarmos o processo do defrag.exe criado pelo Injector.

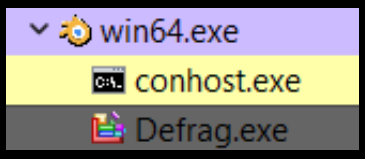

Figura 3 - Árvore de Processos

Tendo identificado um *DOS Header* em memória, é importante validarmos se este cabeçalho se trata do *defrag.exe*, que será executado após a criação do processo. Porém, como é possível observarmos na imagem abaixo, o DOS Header do defrag.exe contém apenas 'MZ' (4d 5a 90), enquanto o cabeçalho identificado no registrador RBP contém 'MZx'.

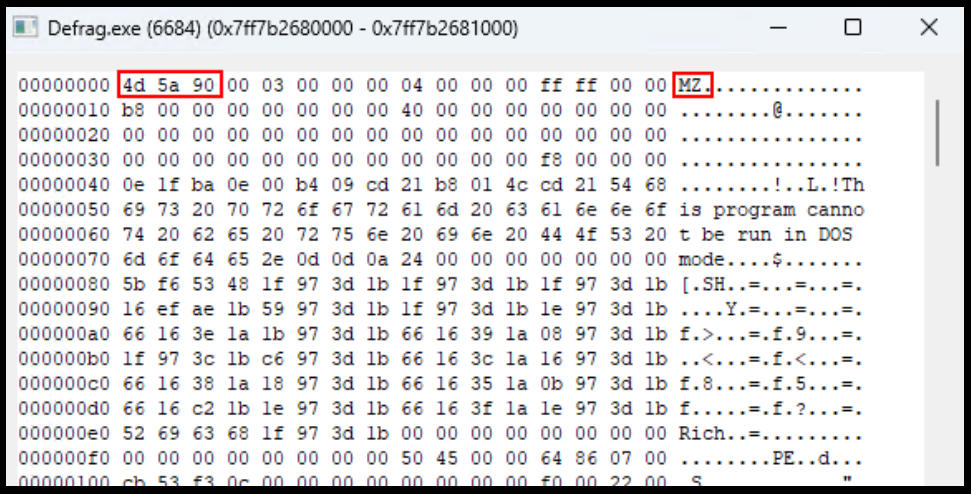

Figura 4 - DOS Header do Executável Defrag

Ao analisarmos o conteúdo presente no registrador RBP, é possível observarmos que de fato, o registrador aponta para um outro executável, diferente do Injector e do defrag.exe. Abaixo, podemos identificar a sequência de bytes do *DOS Header*, que difere do binário que será executado pelo Injector ('MZx' – 4d 5a 78).

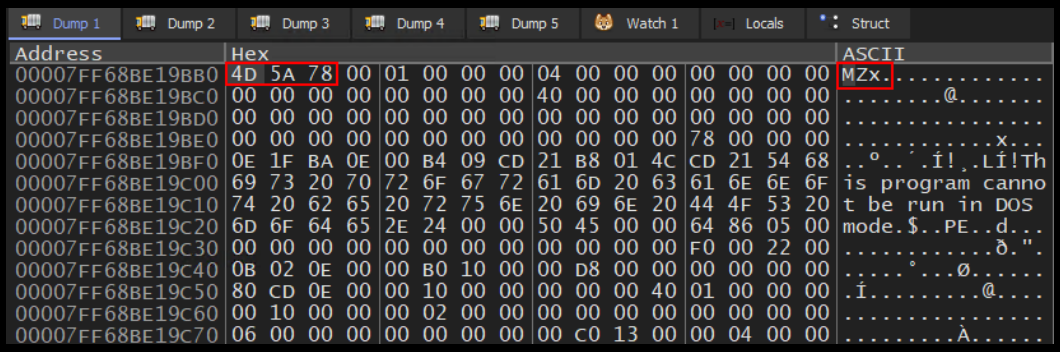

Figura 5 - Identificação do DOS Header do Lockbit5.0 Descriptografado

Ao analisarmos de maneira mais profunda a memória do processo do Injector, é possível encontrarmos de fato um novo executável no endereço especificado no registrador RBP. Este executável é o Lockbit5.0 descriptografado.

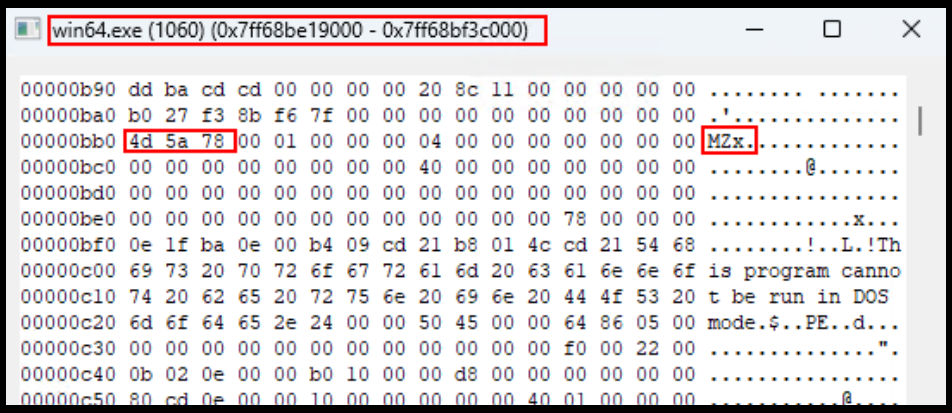

Figura 6 - Identificação do Lockbit5.0 Descriptografado na Memória do Injector

Após criar o processo do defrag.exe, o Injector irá injetar o Lockbit5.0 descriptografado em um endereço remoto do processo defrag.exe, por meio da execução da *Syscall* ZwWriteVirtualMemory. Na imagem abaixo, é possível identificarmos alguns argumentos que são utilizados por esta *Syscall*, como o *Handle* (0xc4) do processo remoto que terá dados injetados na memória, o endereço de memória do processo remoto (defrag.exe) onde será injetado o Lockbit5.0, e o *Buffer*, contendo o endereço para os dados que serão injetados.

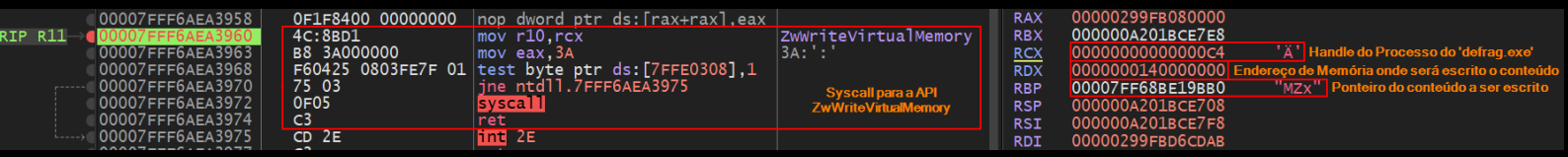

Figura 7 - Injeção do Lockbit5.0 Descriptografado no Processo Defrag

Na imagem abaixo, é possível validarmos o valor (0xc4) do *Handle* do processo alvo, defrag.exe, obtido pelo processo do Injector, que será utilizado como argumento da *Syscall* ZwWriteVirtualMemory, durante o processo de injeção.

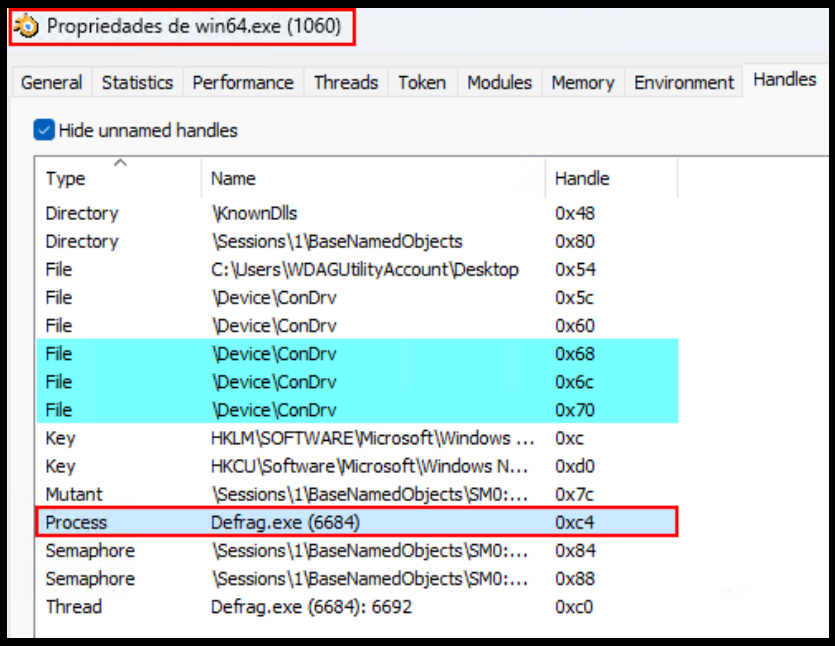

Figura 8 - Identificação do Handle Obtido pelo Injector do Processo Defrag

Na imagem abaixo, podemos observar que de fato o endereço de memória especificado como argumento, referente ao espaço que receberá o Lockbit5.0 descriptografado, de fato existe e encontra-se vazio antes de receber os dados do processo do Injector.

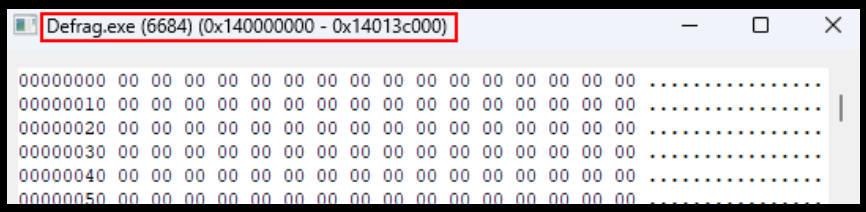

Figura 9 - Espaço de Memória Vazia no Processo do Defrag

Ao executar a *Syscall* ZwWriteVirtualMemory é possível observar o espaço identificado anteriormente, sendo preenchido pelo Lockbit5.0 descriptografado.

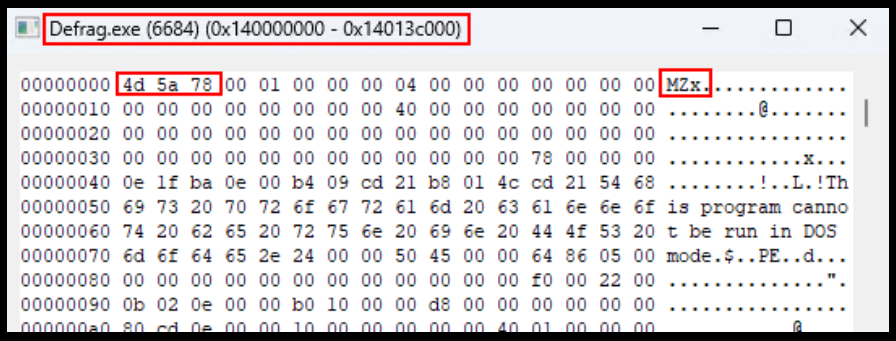

Figura 10 - Lockbit5.0 Injetado em Espaço Anteriormente Vazio do Processo do Defrag

Ao realizar o dump do binário em memória, é possível realizar a comparação referente a presença de determinadas strings críticas para o Lockbit5.0, como o nome do README que será escrito no sistema, que existem no executável extraído da memória, enquanto no Injector não existia.

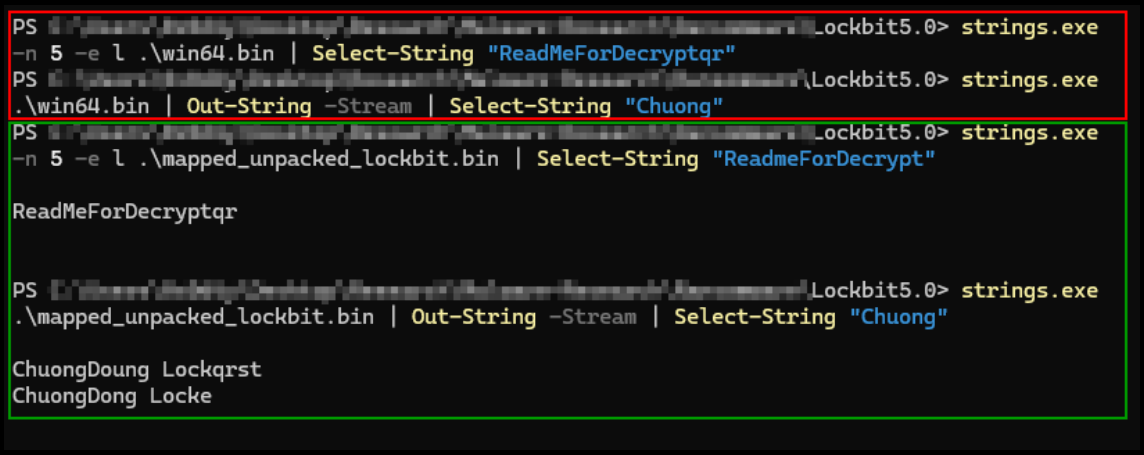

Figura 11 - Validação do Lockbit5.0 Descriptografado Extraído da Memória

ANÁLISE DAS NOVAS PRINCIPAIS CAPACIDADES

Além do processo de *unpacking* via Remote Process Injection, o principal avanço do Lockbit5.0 é a nova rotina customizada de carregamento de DLLs e resolução de APIs de maneira totalmente dinâmica, e envolvendo técnicas híbridas, sendo algumas totalmente customizadas e avançadas, combinadas com técnicas já identificadas anteriormente. Ao ser descriptografado e injetado no processo remoto, a primeira ação do Lockbit5.0 é a resolução de APIs críticas e a implementação de técnicas para evadir produtos de segurança de *Endpoints* como EDRs.

A primeira ação do Lockbit5.0, é mapear DLLs importantes como ntdll.dll, kernel32.dll entre outras, e de maneira manual coletar os endereços das APIs através da técnica *PE Parsing*, que consiste em acessar manualmente as estruturas PE da DLL, em busca dos endereços de determinadas APIs exportadas. Porém, não há strings explícitas de quais APIs será procurada, pois, o Lockbit5.0 implementa um novo algoritmo de *hashing* em múltiplas camadas.

Na imagem abaixo, é possível observarmos parte da rotina de *PE Parsing* das DLLs que serão carregadas durante a execução do Lockbit5.0.

```
dos_header = (struct _IMAGE_DOS_HEADER *)load_pe_func(file_path, 4, 0, 0, file_handle);
v201 = v196;
if ( !dos_header )
{

  v201(module_path);

  v196(v195);
  goto LABEL_75;
}
v276 = (unsigned __int64)v196;
ptr_module_path = module_path;
if ( dos_header->e_magic == IMAGE_DOS_SIGNATURE )
{
  nt_header = (struct _IMAGE_NT_HEADERS64 *)dos_header->e_lfanew;
  if ( (__int64)nt_header < 0x10000001
    && (unsigned __int64)nt_header >= 0x40
    && *(_DWORD *)((char *)&dos_header->e_magic + (_QWORD)nt_header) == IMAGE_NT_SIGNATURE )
  {
    opt_header = (_IMAGE_OPTIONAL_HEADER64 *)((char *)nt_header + (_QWORD)dos_header);
    LODWORD(image_base) = LOWORD(opt_header->ImageBase);
    if ( expected_image_base.m128i_i32[0] == (_DWORD)image_base )
    {
      parse_offset = 0x78;
      if ( !opt_header->DataDirectory[0].Size )
        goto CLEAN_UP;
```

Figura 12 - Código Parcial da Rotina de PE Parsing

Ao acessar manualmente a estrutura IMAGE_EXPORT_DIRECTORY de cada DLL, por meio do *PE Parsing*, o Lockbit5.0 coleta informações referente a APIs exportadas pela DLL, e então implementa um algoritmo customizado de *Hashing* em múltiplas camadas para cada entrada, até encontrar as APIs desejadas. Esta técnica aumenta o nível técnico de desenvolvimento de malware, e consequentemente de engenharia reversa.

Na sequência de imagens abaixo, é possível observarmos alguns trechos da implementação em camadas do algoritmo customizado de *Hashing*.

```
if ( !v1009_1 )
{
  strcpy((char *)&module_str, "user32.dll");
  module_hash = lb5_hash_module(&module_str, module_str_6);
  if ( module_hash && (api_hash = lb5_resolve_export_by_hash(module_hash, 0x66216021)) != 0 )
  {
    ptr_api_hash = api_hash;
    v1009_1 = ((__int64 (__fastcall *)(__int64))sub_140101DE0)(api_hash);
    if ( !v1009_1 )
      v1009_1 = ptr_api_hash;
    sub_140026CA0(0x66216021u, v1009_1);
  }
  else
  {
    v1009_1 = 0;
  }
}
```

Figura 13 - Identificação Parcial da Rotina de API Hashing

```
lb5_hash_scrub_state(&pe_addr_functions[2]);
lb5_hash_fold128(&pe_addr_functions[2], &si128_1);
if ( !IMAGE_EXPORT_DIRECTORY->NumberOfNames )
  return 0;
pe_addr_functions[0] = (IMAGE_EXPORT_DIRECTORY *)((char *)pe_addr_functions[0]
                                                + (((unsigned __int64)v219 << 0x38)
                                                 | ((unsigned __int64)v220 << 0x30)
                                                 | ((unsigned __int64)v221 << 0x28)
                                                 | ((unsigned __int64)v222 << 0x20)
                                                 | (v223 << 0x18)
                                                 | (v224 << 0x10)
                                                 | (v225 << 8)
                                                 | (unsigned __int8)(dll_base_1 ^ v234)));
AddressOfNames += ((unsigned __int64)v73 << 0x38)
                | ((unsigned __int64)v72 << 0x30)
                | ((unsigned __int64)v71 << 0x28)
                | ((unsigned __int64)v70 << 0x20)
                | (v216 << 0x18)
                | (v217 << 0x10)
                | (v218 << 8)
                | (unsigned __int8)(dll_base_1 ^ v69);
v86 = BYTE4(pe_addr_functions[1]);
v234 = (((unsigned __int64)(unsigned __int8)(v227 ^ si128_1.m128i_i8[7]) << 0x38)
      | ((unsigned __int64)(unsigned __int8)(v228 ^ si128_1.m128i_i8[6]) << 0x30)
      | ((unsigned __int64)(unsigned __int8)(v229 ^ si128_1.m128i_i8[5]) << 0x28)
      | ((unsigned __int64)(unsigned __int8)(v226 ^ si128_1.m128i_i8[4]) << 0x20)
      | ((unsigned __int8)(v213 ^ si128_1.m128i_i8[3]) << 0x18)
      | ((unsigned __int8)(v214 ^ si128_1.m128i_i8[2]) << 0x10)
      | ((unsigned __int8)(BYTE4(pe_addr_functions[1]) ^ si128_1.m128i_i8[1]) << 8)
      | (unsigned __int8)(dll_base_1 ^ si128_1.m128i_i8[0] ^ v85))
     + IMAGE_EXPORT_DIRECTORY->AddressOfNameOrdinals;
v87 = 0;
while ( 1 )
{
  *(double *)si128_26.m128i_i64 = lb5_seed_prng_init(0x1DC7, 0xDA50A231LL);
  si128_1 = si128_26;
  si128_27 = si128_26;
  pe_addr_functions[2] = (IMAGE_EXPORT_DIRECTORY *)lb5_seed_prng_emit(&si128_27, 0xDA50A231LL);
```

Figura 14 - Pseudocódigo Parcial da Rotina de Hashing

Ao encontrar as APIs críticas para as rotinas que serão executadas a seguir, o Lockbit5.0 checa se estas APIs se encontram *Hooked* por produtos de segurança como *EDRs* ou *XDRs*. Ao identificar tal característica presente nas APIs críticas, o Lockbit5.0 implementa um *patch* para realizar o *Unhooking* destas APIs.

A identificação de um *Hook* em determinada API, se dá ao identificar que o primeiro byte do *offset* da API a ser chamada, encontra-se o byte 0xE9. Este byte corresponde ao *opcode* JMP, sendo uma instrução de alteração incondicional do fluxo de execução em *Assembly*. Os produtos de segurança avançados, implementam esta técnica (API Hooking), com o objetivo de ao ser chamada uma API que geralmente os Malwares utilizam, o produto de segurança altera os primeiros bytes da chamada da API, adicionando o primeiro byte o *opcode* 0xE9, ou seja um salto para uma outra função controlada pelo produto de segurança, que irá executar uma rotina de análise do conteúdo dos argumentos passados a API Hooked, com o objetivo de identificar comportamentos maliciosos, e por fim, tomar medidas de resposta a uma possível ameaça, se identificada.

Esta técnica é muito utilizada por Malwares Bancários, para coletar informações bancárias via API Hooking, de determinadas APIs utilizadas por navegadores ou softwares de Bancos. Porém, ao identificarem que os produtos de segurança estão implementando esta técnica, os desenvolvedores de *malware* descobriram como executar uma contramedida a técnica de API Hooking por parte dos produtos de segurança, implementando uma rotina manual de identificação de Hooks, e ao serem identificados, aplica-se um *patch* para retirar o Hook (Unhooking).

```
        }
        api_resolved_addr = lb5_resolve_api_by_hash(image_nt_header, export_hash);
        if ( api_resolved_addr )
        {
          first_api_opcode = *api_resolved_addr;
          if ( (first_api_opcode == 0xE9
             || first_api_opcode == 0xFF && api_resolved_addr[1] == 0x25)
            && (*patched_export_addr != (_BYTE)first_api_opcode
             || patched_export_addr[1] != api_resolved_addr[1]) )
          {
            old_protect = 0;
            new_protect = 0;
            addr = api_resolved_addr;
            if ( ptr_VirtualProtect(
                   api_resolved_addr,
                   0x40u,
                   PAGE_EXECUTE_READWRITE,
                   &old_protect) < 0 )
              break;
            ptr_api_resolved_addr = addr;
            *((_WORD *)addr + 4) = *((_WORD *)patched_export_addr + 4);
            *ptr_api_resolved_addr = *(_QWORD *)patched_export_addr;
            if ( ptr_VirtualProtect(
                   ptr_api_resolved_addr,
                   0x40u,
                   old_protect,
                   (unsigned int *)&new_protect) < 0 )
              break;
          }
        }
      }
      --counter;
      ++ptr_rva;
      ++ptr_name_ordinals;
    }
    while ( counter );
  }
  goto CLEAN_UP;
```

Figura 15 - Rotina de API Unhooking para Evadir Hooks de Produtos de Segurança de Endpoints

Ao garantir que as APIs críticas estão *Unhooked*, Lockbit5.0 implementa as técnicas já identificadas em versões anteriores, e que são comuns de famílias de Ransomware.

Uma das técnicas Evasão de Defesas introduzida na versão 4.0 e implementada novamente nesta nova versão, é a desabilitação dos logs por meio de ETW Patching, que podemos observar no pseudocódigo devidamente comentado para melhor compreensão abaixo. No pseudocódigo, podemos observar a identificação de uma variável que recebe o valor em hexadecimal, referente ao *opcode* (0xc3) RET, na qual em Assembly instrui a *CPU* a finalizar a função.

Local cross references to ret_opcode_patch

Xref	Line	Columı	Pseudocode line
w	3684	2	ret_opcode_patch = 0xC3; // Variable Declaration with 0xc3 (ret Asm opcode) to Patch ETW
o	3984	7	&ret_opcode_patch, // 0xC3 -> Assembly Opcode RET

Figura 16 - XRefs de Variável Contendo o Opcode RET

Como é possível observarmos, além da declaração da variável com o valor referente ao *opcode* RET, essa variável também é utilizada efetivamente no processo de ETW Patching, onde o adversário utiliza a API (*após realizar o processo de resolução descrito anteriormente*) WriteProcessMemory para escrever um único byte (0xc3) no início do endereço da API EtwEventWrite, finalizando precocemente a função de registro de logs, sempre que esta API for chamada. Desta forma, nenhum é registrado após a execução desta rotina.

```
_mm_storel_ps((double *)&ptr_WriteProcessMemory, (__m128)si128);//
                                                //
                                                // Rotina de Implementação de ETW Patching
                                                //
((void (__fastcall *)(unsigned __int64, __int64, char *, __int64, __m128i *))ptr_WriteProcessMemory)(
  0xFFFFFFFFFFFFFFFFuLL,                        // hProcess -> Handle do Próprio Processo (0xFFFFFFFFFFFFFFFF)
  lpBaseAddress,                                // Offset da API EtwEventWrite
  &ret_opcode_patch,                            // 0xC3 -> Assembly Opcode RET
  1,                                            // nSize -> Tamanho de Apenas 1 Byte (0xc3)
  &lpNumberOfBytesWritten);
```

Figura 17 - Implementação da Técnica de Evasão de Defesas: ETW Patching

Ao implementar todas as capacidades para Evasão de Defesas, o Lockbit5.0 finalmente implementa as rotinas comuns para uma família de Ransomware, de criptografia de arquivos do sistema operacional, e criação dos arquivos READMEs. Abaixo, é possível observarmos o README completo do Lockbit5.0, que é escrito no sistema vítima.

```
~~~ You have been attacked by LockBit 5.0 - the fastest, most stable and immortal ransomware since 2019 ~~~~

>>>>> You must pay us.

Tor Browser link where the stolen infortmation will be published:
http://lockbitapt67g6rwzjbcxnww5efpg4qok6vpfeth7wx3okj52ks4wtad.onion

>>>>> What is the guarantee that we won't scam you?
We are the oldest extortion gang on the planet and nothing is more important to us than our reputation. We are not a politically motivated group and want nothing but financial rewards for our work. If we defraud even one client, other
clients will not pay us. In 5 years, not a single client has been left dissatisfied after making a deal with us. If you pay the ransom, we will fulfill all the terms we agreed upon during the negotiation process. Treat this situation simply
as a paid training session for your system administrators, because it was the misconfiguration of your corporate network that allowed us to attack you. Our pentesting services should be paid for the same way you pay your system
administrators' salaries. You can get more information about us on wikipedia https://en.wikipedia.org/wiki/LockBit

>>>>> Warning! Do not delete or modify encrypted files, it will lead to irreversible problems with decryption of files!

>>>>> Don't go to the police or the FBI for help and don't tell anyone that we attacked you. They will forbid you from paying the ransom and will not help you in any way, you will be left with encrypted files and your business will die.

>>>>> When buying bitcoin, do not tell anyone the true purpose of the purchase. Some brokers, especially in the US, do not allow you to buy bitcoin to pay ransom. Communicate any other reason for the purchase, such as: personal investment
in cryptocurrency, bitcoin as a gift, paying to buy assets for your business using bitcoin, cryptocurrency payment for consulting services, cryptocurrency payment for any other services, cryptocurrency donations, cryptocurrency donations
for Donald Trump to win the election, buying bitcoin to participate in ICO and buy other cryptocurrencies, buying cryptocurrencies to leave an inheritance for your children, or any other purpose for buying cryptocurrency. Also you can use
adequate cryptocurrency brokers who do not ask questions for what you buy cryptocurrency.

>>>>> After buying cryptocurrency from a broker, store the cryptocurrency on a cold wallet, such as https://electrum.org/ or any other cold cryptocurrency wallet, more details on https://bitcoin.org By paying the ransom from your personal
cold cryptocurrency wallet, you will avoid any problems from regulators, police and brokers.

>>>>> Don't be afraid of any legal consequences, you were very scared, that's why you followed all our instructions, it's not your fault if you are very scared. Not a single company that paid us has had issues. Any excuses are just for
insurance company to not pay on their obligation.

>>>>> You need to contact us via TOR sites with your personal ID

Download and install Tor Browser https://www.torproject.org/
Write to the chat room and wait for an answer, we'll guarantee a response from us. If you need a unique ID for correspondence with us that no one will know about, ask it in the chat, we will generate a secret chat for you and give you ID
via private one-time memos service, no one can find out this ID but you. Sometimes you will have to wait some time for our reply, this is because we have a lot of work and we attack hundreds of companies around the world.

Tor Browser link for chat with us:
http://lockbitsuppyx2jegaoyiw44ica5vdho63m5ijjlmfb7omq3tfr3qhyd.onion

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>> Your personal identifier to communicate with us ID: [redacted] <<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

>>>>> Advertising:
Want a lamborghini, a ferrari and lots of titty girls? Sign up and start your pentester billionaire journey in 5 minutes with us.
http://lockbitfbinpwhbyomxkiqtwhwiyetrbkb4hnqmshaonqxmsrqwg7yad.onion
After registration, you will receive the most flawless and reliable tools for encrypting almost all operating systems on the planet and a platform for negotiating with attacked companies.

Version: ChuongDong v1.01 | x64
```

Figura 18 - README do Lockbit5.0

CONCLUSÃO

Esta nova versão do Lockbit de fato trouxe atualizações relevantes. Das principais atualizações citadas ao longo da análise, eu gostaria de ressaltar a técnica de *resolução dinâmica em múltiplas camadas das APIs* e o processo de *Unpacking* por meio da *Injeção em Processo Remoto*. Ambas as implementações técnicas, são novidades não vistas anteriormente, e que de fato aumentou o nível técnico de desenvolvimento, e de análise da amostra.

Também é importante ressaltar, que houve uma diminuição de capacidades implementadas pelo Lockbit, mantendo-se apenas as capacidades úteis para evadir produtos de segurança de Endpoint, demonstrando portanto, a preocupação em evadir determinadas proteções presentes nos sistemas alvos. Apesar da implementação das técnicas Anti-Forense, as principais TTPs que caracterizam uma infecção de Ransomware são identificáveis, se os sistemas infectados fizeram parte de um escopo de monitoramento estruturado por meio de SOC.

MAPEAMENTO MITRE ATT&CK

Abaixo segue o mapeamento do *MITRE ATT&CK*, referente aos comportamentos produzidos pelo *Lockbit5.0* Ransowmare.

Tactic	Technique	ID
Defense Evasion	Impair Defenses: Indicator Blocking	T1562.006
	Indicator Removal: Clear Windows Event Logs	T1070.001
	Impair Defenses: Disable or Modify Tools	T1562.001
	Dynamic API Resolution	T1027.007
	Portable Executable Injection	T1055.002
Impact	Internal Defacement	T1491.001
	Inhibit System Recovery	T1490
	Data Encrypted for Impact	T1486

Tabela 2 - Mapeamento MITRE ATT&CK do Lockbit5.0

MAPEAMENTO MALWARE BEHAVIOR CATALOG (MBC)

O Lockbit5.0 Ransomware implementa um conjunto de técnicas amplamente reconhecidas no framework Malware Behavior Catalog, no qual é possível identificar as capacidades identificadas durante a análise e implementadas pela amostra.

Tactic	Technique	ID
Cryptography	Encrypt Data	C0027
Discovery	File and Directory Discovery	E1083
	System Information Discovery	E1082
Defense Evasion	Disable or Evade Security Tools	F0004
	Self Deletion	F0007
	Process Injection	E1055
Anti-Static Analysis	API Hashing	B0032.001
	Import Address Table Obfuscation	B0032.011
File System	Read File	C0051
	Delete File	C0047
	Writes File	C0052
	Create Process	C0017
	Create Thread	C0038
Impact	Data Encrypted for Impact	E1486

Tabela 3 - Mapeamento MBC do Lockbit5.0

INDICADORES DE COMPROMETIMENTO

Aqui você encontrar os indicadores de comprometimento coletados, referente ao Lockbit5.0.

Identificador de Hash	
md5	95daa771a28eaed76eb01e1e8f403f7c
sha1	cdd5717fd3bfd375c1c34237c24073e92ad6dccc
sha256	7ea5afbc166c4e23498aa9747be81ceaf8dad90b8daa07a6e4644dc7c2277b82
File Name	Blender.exe

Tabela 4 - Indicadores de Comprometimento

Tipo de Indicador	Indicador	Função/Característica
Data Leak Site	http://lockbitfbinpwhbyomxkiqtwhwiyetrbkb4hnqmshaonqxmsrqwg7yad.onion	DLS do Lockbit5.0
Chat Site	http://lockbitsuppyx2jegaoyiw44ica5vdho63m5ijjlmfb7omq3tfr3qhyd.onion	Tox Chat Site

Tabela 5 - Indicadores de Comprometimento de Rede

REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia;**
- **CTI Purple Team *by* ISH Tecnologia;**
- **MITRE ATT&CK;**
- **Malware Behavior Catalog.**

AUTORES

- **Ícaro César – Malware Researcher**

heimdall
security research
A DIVISION OF ISH