BUNNYSHIELD 0.0.1

O BunnyShield é um software desenvolvido para a finalidade de proteger sistemas UNIX de ataques feitos por Ransomware de qualquer família. Para atingir tal objetivo, é utilizado um sistema de monitoramento em tempo-rea que trabalha em conjunto com o sistema de auditoria do Linuxl, escaneando por qualquer atividade maliciosa após algum evento suspeito ocorrer.

1 FUNCIONAMENTO

Como o BunnyShield funciona?

O funcionamento se por um sistema de monitoramento em tempo-real do sistema de arquivos, o qual possui regras de validação para determinar se o evento recente (ou o conjunto de eventos recentes) faz parte da atividade de um processo malicioso dentro em execução.

Validação de eventos

Para entender melhor o funcionamento do BunnyShield, é necessário entrar em detalhe de como cada validação de evento funciona.

A validação de eventos ocorre pelos seguintes critérios:

- Qualquer tipo de edição na HoneyFolder;

- HoneyFile deletado;

- Pasta com HoneyFile foo deletada;

- HoneyFile modificado;

- Arquivo com extensão desconhecida criado;

- Diversos eventos no sistema de arquivos foi detectado.

Criação da HoneyFolder

Como dito acima, o BunnyShield se utiliza de uma HoneyFolder para ser um ponto chave na detecção de atividade maliciosa no sistema.

Basicamente, a HoneyFolder é uma pasta com um único intuito: servir como uma armadilha para um Ransomware.

Uma vez que estes malwares costumam dar prioridade para criptografar a partir de uma pasta base como a /home), a HoneyFolder é criada neste diretório, onde dentro dela há 10.000 PDFs com nomes extremamente sedutores para o Ransomware, como "secret", "bank", "credit-card", "data", "password", "finantial", "money", "personal", "paypal", e "credential".

Com isto, como a HoneyFolder é simples e unicamente uma pasta com objetivo de detectar atividade maliciosa, nenhum processo ou usuário terá interesse em fazer qualquer modificação nela, ao menos que, claramente, seja um Ransomware.

Criação dos HoneyFiles

Fora a HoneyFolder, o BunnyShield também conta com um robusto sistema de HoneyFiles espalhados por diretórios que o usuário tenha interesse de monitorar para proteger seus arquivos. A criação e funcionalidade dos HoneyFiles é mais complexo, portanto, é necessário explicar esta função mais detalhadamente.

Primeiramente, para ser possível obter os dados do processo malicioso fazendo modificações no sistema de arquivos, é utilizado o sistema de auditoria do Linux, o qual é extremamente modular, permitindo a criação de diversas regras personalizadas para criar logs de eventos ocorridos no sistema de arquivos ou mesmo de chamadas de sistema.

Com isso dito, antes da própria criação dos HoneyFiles, é criado uma regra de auditoria para cada diretório que o usuário pretende monitorar por atividade maliciosa. Esta regra de auditoria será imprescindível para posteriormente encontrar o processo realizando mudanças no sistema de arquivos e validar se o mesmo é ou não malicioso.

Além das regras de auditoria criadas para cada diretório, também é criada uma regra de auditoria para monitorar chamadas de sistema "execve" para o caminho "/bin/sh", ou, em outras palavras, para logar toda vez que um processo abrir uma nova shell. Isto é feito para detectar Ransomwares mais poderosos que conseguem criar um processo ou thread para cada arquivo sendo criptografado. Vale destacar que esta regra em específico não funciona em dependência de nenhum diretório sendo monitorado (ela não monitora o sistema de arquivos, e sim as chamadas de sistema "execve" no caminho citado anteriormente).

Após a criação das referidas regras de auditoria, o processo de geração de HoneyFiles iniciará e, para garantir que seja utilizada a menor quantidade possível de HoneyFiles com a maior eficiência possível, deverão ser atendidos certos requisitos para que um HoneyFile seja criado. Apenas será criado caso:

- O diretório atual é o diretório base;

- O diretório atual é um dos diretórios após o diretório base;

- O diretório atual não possui mais pastas dentro dele.

Através dessa validação, é torna-se possível criar os HoneyFiles apenas nos diretórios base e nos diretórios mais profundos presentes no diretório base, evitando que um HoneyFile seja criado por pasta, o que seria desnecessário visto o corportamento de Ransomwares, os quais começam criptografar de cima para baixo ou de baixo para cima, e nunca do meio. Portanto, independentemente de onde o Ransomware começar sua atividade maliciosa, ele irá esbarrar em um HoneyFile.

Já se tratando da estrutura dos HoneyFiles em si, eles basicamente são arquivos ocultos com um prefixo "0-secret-bsfile-", uma string aleatória de 25 digitos alfanuméricos, e no fim uma extensão .txt. Por conta de ser um arquivo oculto e que inicia com o dígito "0", ele sempre estará no topo do diretório, fazendo com que o Ransomware acabe começando a criptografar um HoneyFile antes de tocar qualquer outro arquivo.

Dentro de um HoneyFile é possível encontrar algo semalhante ao texto abaixo:

THIS IS A BUNNYSHIELD FILE! PLEASE, DO NOT MOVE, DELETE, RENAME OR MODIFY THIS FILE! YOU ARE STILL FREE TO MOVE ANY FOLDER THAT CONTAINS A FILE LIKE THIS.

Credit card details: 4504569356688550007630139451999663472303162821823 3

É possível perceber que todo recurso para tornar o HoneyFile mais atrativo para um Ransomware é utilizado, uma vez que o nome do arquivo contém a palavra "secret", e dentro dele ainda possui uma frase dizendo "Credit card details" com um número aleatório gerado ao lado. Basicamente um Ransomware que utiliza de técnicas de validação para determinar a prioridade de quais arquivos criptografar seria facilmente enganado.

Por fim, com o HoneyFile criado, será criado um dicionário em Python que conterá o caminho absoluto do HoneyFile e uma hash SHA1 gerada com base nos bytes presentes do HoneyFile. Este dicionário ficará salvo na memória para, após a criação de todos os HoneyFiles, um arquivo .JSON com todas as entradas de cada HoneyFile ser gerado, o qual será indispensável para detectar modificações nos HoneyFiles, assim como para atualizar cada diretório, criando ou removendo HoneyFiles caso seja o caso. Isto será explicado mais a frente. Além do .JSON, também é criado um arquivo .TXT que possui todos os nomes únicos de cada HoneyFile.

Remoção dos HoneyFiles e da HoneyFolder

Caso o usuário queira deletar os HoneyFiles de determinado diretório, assim como a HoneyFolder, o BunnyShield possui uma opção para remover automaticamente os mesmos.

Atualização do arquivos .JSON e .TXT dos HoneyFiles

Não faz o menor sentido possui um sistema estático de HoneyFiles, isto é, um sistema que cria HoneyFiles em um diretório e coloca todas suas informações em um JSON e um TXT, mas não atualiza ambos caso haja alguma modificação no sistema de arquivos, como por exemplo uma pasta com HoneyFile sendo movida de um lugar para outro, uma nova pasta sendo criada, uma pasta com várias pastas com HoneyFiles sendo deletada etc.

O BunnyShield é um software extremamente modular que possui um sistema robusto para atualizar o sistema de arquivos para a criação ou remoção de HoneyFiles, onde tudo acontece de forma automática, sendo gerenciado 100% pelo software, fazendo com que o usuário não precise de importar com, por exemplo, estar deletando ou movendo uma pasta com HoneyFiles, prejudicando o funcionamento do sistema de monitoramento e detecção do BunnyShield.

Qualquer modificação feita no sistema de arquivos que crie uma pendência para criar, atualizar, ou deletar um HoneyFile será feita de forma automática, e os dados presentes no .JSON e no .TXT serão atualizados.

Sistema de monitoramento em tempo-real

Aqui é onde o funcionamento prático de validação e detecção de atividade maliciosa efetivamente começa.

O BunnyShield funciona através de "observers", que são monitores de diretórios do módule "watchdog" do Python. Com esta ferramenta, é possível detectar eventos de criação, movimentação, modificação ou remoção de arquivos. Aqui, todos os eventos citados na seção "validação de eventos" são validados para ativar ou não o escaneador de Ransomware.

Ainda, é através deste sistema de monitoramento em tempo-real é possível atualizar o sistema de arquivos para a criação ou remoção de HoneyFiles (assim como .JSON e o .TXT, respectivamente).

Uma vez que algum evento que ativa o escaneador de Ransomware acontece, a o processo de detecção e validação de um processo malicioso começa a rodar.

Sistema de detecção e validação de Ransomwares

Uma vez que este sistema entra em ação, ele irá tentar detectar e finalizar um processo malicioso por um método ou outro: edições realizadas no sistema de arquivos ou chamadas de sistema "execve" no caminho "/bin/sh". Ambos métodos

fazem uso dos logs do sistema de auditoria do Linux, os quais trazem informações vitais para que o processo de validação ocorra.

Primeiramente, será puxado os últimos 10 logs do sistema de auditoria do Linux com a key do BunnyShield (para filtrar resultados). Um log terá a seguinte estrutura:

time->Wed Sep 28 19:31:43 2022

type=PROCTITLE msg=audit(1664404303.272:1054236): proctitle=707974686F6E33006261736963726172616E736F6D776172652E7079002D70002F686F6D652F6D6174686575736865696465 6D616E6E2F446F63756D656E74732F4769746875622F4368616C6C656E67652F7765627365 6974652D746573742F72616E736F6D776172652D746573742F656E63727970742D74657374002D65

type=PATH msg=audit(1664404303.272:1054236): item=1 name="/home/matheusheidemann/Documents/Github/Challenge/website-test/ransomware-test/encrypt-test/folder4/subfolder1500/file2.txt" inode=5900566 dev=08:01 mode=0100644 ouid=1000 ogid=1000 rdev=00:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0

type=PATH msg=audit(1664404303.272:1054236): item=0 name="/home/matheusheidemann/Documents/Github/Challenge/website-test/ransomware-test/encrypt-test/folder4/subfolder1500/" inode=5898271 dev=08:01 mode=040755 ouid=1000 ogid=1000 rdev=00:00 nametype=PARENT cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0

type=CWD msg=audit(1664404303.272:1054236): cwd="/home/matheusheidemann/Documents/Github/Challenge/website-test/ransomware-test/B4S1CR4NS0MW4R3"

type=SYSCALL msg=audit(1664404303.272:1054236): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=7f6fa2fecc30 a2=80241 a3=1b6 items=2 ppid=455665 pid=455666 auid=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts9 ses=6 comm="python3" exe="/usr/bin/python3.9" subj==unconfined key="bs-file-event"

Através destes logs é possível iniciar uma validação para definir se o processo sendo escaneado é ou não um falso-positivo. Por exemplo, no campo "tty" do log acima é possível identificar o valor "pts9", o que significa que o processo está se utilizando de algum terminal do Linux para executar sua atividade maliciosa. Isso já elimina qualquer possibilidade de um gerenciador de arquivos como o Nautilus ou Thunar de serem tratados como processos maliciosos, uma vez que ele não possuem um tty associado a eles (isso evita que o processo do gerenciador de arquivos seja finalizado pelo BunnyShield, caso o usuário esteja, por exemplo, deletando milhares de arquivos).

Também é possível perceber no log a informação do PID e do PPID do processo, sendo a peça essencial para continuar a validação do processo.

Com o PID em mãos, será criada uma nova thread para um função com a finalidade de validar se o PID possui traços de comportamento suspeito.

Para determinar se o processo é ou não suspeito, usando o strace, serão analisadas as chamadas de sistema que este está fazendo, levando em conta a quantidade de certas chamadas de sistema em 1 segundo. As chamadas de sistema usadas para a validação são: lseek, openat, open, close, read, write, epool_ctl, unlink, unlinkat. Ainda, a validação de certas chamadas de sistema não ocorre de forma redundante (ex: caso já haja diversas chamadas de "open", mas também hajam várias de "openat", apenas uma das duas será utilizada para validação) assim evitando problemas com falsos-positivos.

Após validar as chamadas de sistema, também haverá uma validação da quantidade de bytes que o processo escreveu em 1 segundo. Caso este seja maior que 1.000.000 bytes, será adicionado outro ponto de flag suspeita.

Uma vez que o número de flags suspeitas é atingido, o processo analisado é colocado como malicioso, e a funcionalidade para finalizar este e impedir que o mesmo continue a funcionar entrará em ação.

Nesta atividade, o BunnyShield usará de várias validações para conseguir o caminho do arquivo que o processo malicioso está exeutando e também pegará tanto o PID quanto o PPID deste para que sejam finalizados.

Porém, não faria o menor sentido finalizar um processo malicioso sem antes impedir que o mesmo seja executado novamente, portanto, neste processo o BunnyShield tenta pegar o caminho do diretório do qual o Ransomware está executando seu arquivo e bloqueia qualquer direito de execução, dessa forma, quando o processo malicioso for finalizado, ele não conseguirá se executar novamente de forma automática, e assim, o arquivo do Ransomware será deletado e então o permissionamento original do diretório será reestabelecido (caso seja possível encontrar o arquivo do Ransomware).

Com isso, o BunnyShield não só parou a atividade do Ransomware em poucos segundos, como também limpou o sistema do arquivo malicioso do mesmo, impedindo uma reinfecção pós a finalização do Ransomware.

Detecção de keys de criptografia

Uma vez que o BunnyShield possui um sistema de monitramento em tempo-real para o sistema de arquivos, é possível captar qualquer novo arquivo sendo criado no mesmo, e assim, caso um Ransomware crie sua chave de criptografia no sistema (seja por criptografia simética, assimétrica, ou ambas) o BunnyShield estará atento para copiar todas as informações deste arquivo e criar um cópia da chave de criptografia em uma pasta protegida. Uma vez que a chave de criptografia está em mãos, o usuário não só está livre a atividade do Ransomware, mas qualquer arquivo criptografado no processo pode ser facilmente recuperado.

2. CONFIGURAÇÕES

O BunnyShield é um software extremamente modular e que permite diversas configurações pelo usuário, sendo elas:

honeyfile-json-alias: o nome do arquivo .JSON com o caminho absolute a hash de cada HoneyFile;

honeyfile-txt-alias: o nome do arquivo .TXt com o nome de cada HoneyFile;

file-event-rule-name: o nome da regra de auditoria do BunnyShield para logs no sistema de arquivos;

file-event-open-shell-name: o nome da regra de auditoria do BunnyShield para logs no de chamadas de sistema "execve" no caminho "bin/sh";

action: a ação a ser tomada com os HoneyFiles (criar ou deletar);

path-to-honeyfolder: o caminho para o diretório da HoneyFolder;

path-to-whitelistedfolder: o caminho para o diretório da Whitelisted Folder;

directories: os diretórios monitorados;

honeyfile-prefix: o prefixo de texto para os HoneyFiles;

skip-to-monitor: pular uma nova criação de HoneyFiles e HoneyFolder e simplesmente iniciar o monitor em tempo-real.