GARF: Prevenção contra ataques usando Honeypots

Trabalho de Conclusão do Curso de
Tecnologia em Sistemas para Internet

Rafael de Oliveira Queiroz
Orientador(a): César Augusto Hass Loureiro

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Campus Porto Alegre
Av. Cel Vicente, 281 - Porto Alegre – RS – Brasil

r.queiroz@hotmail.com.br, cesar.loureiro@poa.ifrs.edu.br

Resumo. *Ataques contra servidores e serviços na internet são cada vez mais frequentes, necessitando de medidas preventivas cada vez mais pró-ativas. Assim, este trabalho propõe contribuir com a área de segurança da informação através do desenvolvimento de um gerador automatizado de regras em Firewalls usando históricos de acessos a Honeypots.*

1. Introdução

A *internet* nunca foi um lugar seguro. Nas últimas duas décadas presenciamos um desenvolvimento rápido de novas tecnologias e, consequentemente, um rápido crescimento em crimes cibernéticos [KIZZA 2015]. Garantir a segurança de nossas informações tem se tornado uma tarefa cada vez mais complexa. Quando nossas informações eram armazenadas em papéis, bastava guardá-las em cofres. Hoje, com todas as soluções modernas de armazenamento, precisamos de soluções mais elaboradas de segurança [TCU 2012].

Segundo Stallings (2011), ataque é qualquer ação que possa corromper as informações de uma empresa, como o que ocorreu em 2016 em que fomos vítimas de um ataque de negação de serviço (D.O.S.) de larga escala, afetando diversos serviços como Spotify, Twitter entre outros [PAYÃO 2016]. Um D.O.S. é uma técnica maliciosa que tem como objetivo tornar recursos de um sistema indisponível. Comparando 2016 com 2017, na figura 1, é possível perceber um aumento no número de ataques D.O.S. sofrido por empresa [OSBOURNE 2017]. Para que tenham uma chance maior de eficácia, estes ataques, são realizados através do uso de *botnets*, que refere-se aos conjuntos de máquinas comprometidas com acesso a *internet* tomadas por uma entidade criminosa [GROSS 2015]. Com ameaças como essas, a utilização de mecanismos que possam diminuir nossa insegurança torna-se importantíssimo, como é o caso do *Honeypot*. Desde sua criação, o projeto *Honeypot* tem como propósito a identificação de ataques automáticos. Um servidor *Honeypot* é um recurso de segurança cuja função é ser sondado, atacado e comprometido; ou seja, ele simula serviços falhos, tornando-se vulnerável e consequentemente, facilitando as invasões [SPITZNER 2002]. Com isso, todas as tentativas de acesso em seus serviços, são registradas em arquivos de *logs*.

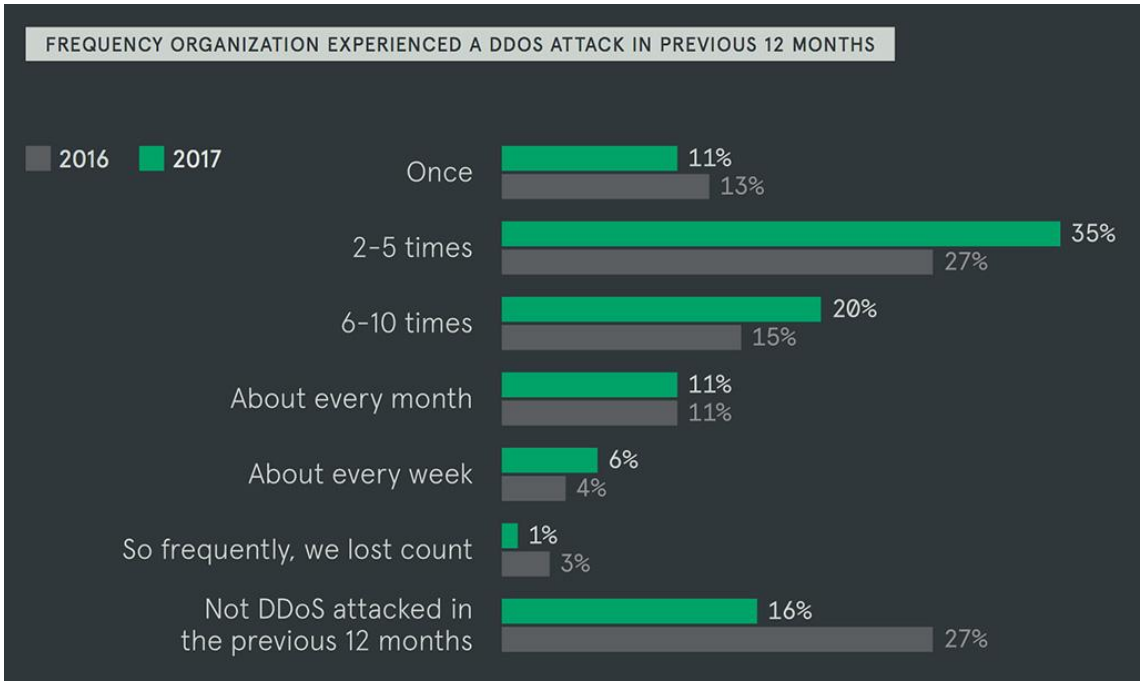

Figura 1. Frequência com que empresas sofreram ataques DDOS no período de 12 meses

Com o intuito de utilizar os registros de *logs* gerados por servidores *Honeypot* e contribuir com a área de segurança da informação, o objetivo deste trabalho é desenvolver um gerador automatizado de regras em Firewall, o GARF, que servirá como uma solução preventiva contra ataques. A ferramenta analisará *logs* de acesso gerados por servidores *Honeypot*, coletando informações como: IP de origem, porta de destino, data de acesso e protocolo e armazená-las em um banco de dados. Ao final deste processo existirá uma base alimentada com informações sobre possíveis ameaças. A partir deste ponto a ferramenta será capaz de agrupar e analisar os registros armazenados e decidir se esses acessos representam alguma ameaça. A análise será baseada no número de tentativas de acesso de um mesmo IP, em uma mesma porta, usando o mesmo protocolo ou apenas por IP. Essa configuração ficará a cargo do usuário. Caso algum atacante seja identificado, o sistema se encarregará de automaticamente criar regras de bloqueio no *firewall* para impedir futuras tentativa de intrusão. O tempo de duração do bloqueio será configurado pelo administrador da solução.

2. Trabalhos Relacionados

Atualmente, no mercado, existem diversas soluções para análise de *logs*. Nesta seção será apresentado um estudo sobre algumas dessas ferramentas. Serão destacados pontos negativos, positivos gerais, relacionados a análise de *logs* e segurança de redes de computadores.

A primeira é a pilha E.L.K. Uma solução *open source* de análise de *logs* composta por três ferramentas altamente configuráveis, são elas: Elasticsearch, Logstash e Kibana. O Elasticsearch é responsável pelo armazenamento de dados e pelo motor de buscas. O

Logstash é responsável por coletar, fazer o *parsing* [1] e filtrar as informações estruturadas, ou não estruturadas. Além disso, ele se encarrega de armazenar em tempo real os dados filtrados no Elasticsearch. Por fim, o Kibana, é uma plataforma cujo objetivo é apresentar através de gráficos e relatórios as informações armazenadas no Elasticsearch. Essa pilha possui a capacidade de analisar uma grande quantidade de *logs* devido a sua arquitetura com escalonamento vertical, ou seja, todo esse trabalho pode ser dividido em diversos nodos [CHNAJED 2015].

A segunda possibilidade é o *Splunk Enterprise*, uma solução comercial de gestão e análise de dados fornecida pela empresa Splunk. Ela é capaz de indexar arquivos de *logs* e torná-los documentos procuráveis, facilitando a busca por qualquer informação disposta dentro destes. Nesta ferramenta não é necessário uma pré-configuração do formato dos arquivos importados, basta que estes sejam arquivos de texto [CARASSO 2012]. Esta abordagem pode ser vantajosa pois facilita a importação de dados. Entretanto, o excesso de informação dispostas pode poluir a visualização dos dados. Dificultando a obtenção de resultados claros, o que não é um ponto positivo.

A terceira opção é uma plataforma de monitoramento de segurança, proposta por Sunil Gupta (2012), no artigo *Logging and Monitoring to Detect Network Intrusions and Compliance Violations in the Environment*. Esta plataforma une informações coletadas por ferramentas de monitoramento de redes e *logs*. Ela foi desenvolvida baseada no *Security Onion*, uma distribuição Linux para detecção de intrusões. O sistema propõe coletar dados de anomalias detectados na rede ou nos *logs* e centralizá-los em um único lugar, facilitando a geração de relatórios e alertas para o usuário [GUPTA 2012].

A quarta ferramenta é Kippo. Esse é um Honeypot projetado para registrar todos os ataques de força bruta e toda a interação do atacante com o servidor após a invasão. Essa ferramenta utiliza um serviço *Honeypot* que simula o acesso via SSH ao servidor. A partir do momento em que o atacante toma controle desta isca, ele começa a ter suas ações monitoradas. O Kippo fornece um sistema de arquivos falso e algumas ferramentas usuais de um servidor Linux, como o wget [2] . Assim é possível analisar o comportamento do invasor durante sua sessão. Todos os dados coletados ficam a disposição do usuário em uma interface web. Essa gera diversos gráficos com informações como: quais senhas foram testadas na tentativa de acesso, quais nomes de usuário foram testados, quais arquivos o intruso baixou durante o acesso, entre outras. [RAWAT 2014]

A última ferramenta analisada foi o fail2ban, uma ferramenta prevenção contra ataques. Ela monitora os *logs* de acesso da máquina em que está instalado e gera regras de bloqueio ao chegar a um certo número de tentativas de acesso por IP. O sistema possui um arquivo onde são configurados quais serviços serão analisados, entre as possibilidades estão: ssh, pam, xinetd, apache, vsftpd, proftpd, wuftpd, postfix, couriersmtp, courierauth, sasl e named; qual o tempo de duração das regras, tempo de execução, informações de e-mail, entre outros parâmetros importantes para o funcionamento da *software*. A manipulação regras é feita através do IPtables. Além de gerar regras de bloqueio, ele possui uma *whitelist*, ou seja, permite que o usuário configure uma lista de IPs que serão poupados das verificações. O sistema também envia notificações por e-email desde que o serviço de SMTP esteja configurado.[ELLINGWOOD 2014] O fail2ban é a aplicação

[1] parsing: processo de transformar dados não estruturados em dados estruturados

[2] wget é uma ferramenta gratuita para downloads não interativos de arquivos. Suporta os protocolos: HTTP, HTTPS e FTP

mais semelhante ao software proposto nesse artigo. A grande diferença entre o fail2ban e o GARF é a origem dos *logs*. Enquanto o fail2ban busca coletar os dados da máquina em que está instalado (e conseguentemente sendo atacada), o GARF se alimenta de *logs* providos por Honeypots, que normalmente são executados em máquinas externas especializadas em coletar dados de invasores.

Para que haja uma visão mais clara dos aspectos principais apresentados pelo GARF e pelos sistemas analisados anteriormente, foi desenvolvida uma tabela comparativa contendo uma lista das aplicações mais relevantes e algumas características importantes para o sucesso da aplicação. Estas informações podem ser visualizadas na Tabela 1.

Tabela 1. Trabalhos relacionados

Característica	ELK	Splunk	Fail2ban	GARF
Open source	x		x	x
Análise de *logs*	x	x	x	x
Geração automatizada de regras			x	x
Pró-atividade				x
Usar Honeypot				x
Interface de Configuração	x			x

Contudo, para que um sistema de prevenção contra ataques seja eficiente, é necessário que decisões sejam tomadas dinamicamente a medida que possíveis intrusos sejam identificados e bloqueados. Essa abordagem é proposta apenas pelo fail2ban. As outras, visam o monitoramento dos servidores, através da análise dos *logs*, gerando alertas e relatórios sobre anomalias detectadas. Deixando a tomada de decisão para o usuário.

3. Referencial Teórico

O GARF está sendo constituído sobre o conceito criado pelo projeto *Honeypot*. Esse é um sistema desenvolvido para enganar e coletar informações. Normalmente são instalados em servidores de *Firewall* [EVEN 2000]. Este propõe um ambiente controlado que simula serviços reais de um servidor. Ele tem como objetivo ser cobiçado, atacado e corrompido [SPITZNER 2002]. Todas as ações decorrente deste processo são gravadas em *logs*. Um *log* é um arquivo de auditoria que registra o histórico de acontecimentos, tornando-se um recurso essencial para sistemas de seguranças como os de detecção

de intrusão [STALLINGS 2011]. Estes registros podem ser tanto nativos ou específicos. Os nativos não possuem um formato definido mas estão presentes em qualquer sistema operacional. Já os específicos, possuem um formato definido e são gerenciados por uma ferramenta [VIDAL 2015]. O que torna o *Honeypot* um recurso interessante de segurança, onde todas informações coletadas por ele podem ser consideradas maliciosas. O *Honeypot* não substitui uma solução de *Firewall*, mas sim trabalha junto a ela [EVEN 2000].

Um *Firewall* é um conjunto de ferramentas, de segurança de rede, podendo ser um *hardware* ou *software*. Ele é definido entre a rede de uma empresa, por exemplo, e outras redes externas, servindo como uma barreira. Nele, é possível configurar regras que definem o que pode, ou não, trafegar para o interior da rede. Existem alguns tipos de *Firewall* que são aplicados em diferentes níveis de camada de uma rede. A ferramenta desenvolvida neste artigo tem como foco a utilização de um *Firewall* de filtro de pacotes (*Netfilter*). Esse é aplicado nas camadas de transporte e aplicação do protocolo TCP/IP e tem como objetivo analisar e filtrar todo os pacotes que passam por essas camadas do protocolo, estejam eles entrando ou saindo da rede. Esta permeabilidade é definida através de regras predefinidas [NETO 2004]. Estas regras são gerenciadas pela ferramenta *IPTables* no sistema operacional Linux.

O IPtables é uma interface para manipulação das tabelas do *Netfilter* que estão incorporadas no *kernel* do Linux. Este possui quatro tabelas importantes para armazenar as regras, são elas:*Filter, NAT, Mangle* e *Raw*. A primeira armazena as regras de filtragem. A segunda é responsável por manipular os endereços *IPs*, utilizando as regras de tradução de pacotes para isso. A terceira é utilizada para alterar as informações dos cabeçalhos de endereçamento, e a última é responsável por marcar pacotes que são de um mesmo estado, no caso de um *Firewall* que mantenha o estado da conexão [ELLINGWOOD 2015]. Diferentemente do *Honeypot* que utiliza *logs* nativos, um *Firewall* utiliza logs específicos. Esses, por sua vez, são estruturados pelo *Syslog*, protocolo que fornece um ponto central de coleta e processamento de *logs*. Os pacotes desse protocolo possuem um tamanho máximo de 1024 *bytes* e carregam as seguintes informações: onde o *log* está sendo gerado, um código de severidade (ex: *Error, Alert, Warning, ...*), data e hora que foram gerados, uma identificação de quem gerou-os e a mensagem a ser relatada [DEVERIYA 2005].

Neste projeto, o gerenciamento dos *logs*, sejam esses nativos ou específicos fica a cargo da pilha *ELK*, que é um conjunto composto por 4 serviços *open source* que visam ajudar nas tarefas de coletar, fazer *parsing*, analisar e armazenar os dados. Os serviços que compõe a pilha ELK são: Logstash, Elasticsearch, Beats e Kibana, o último não será utilizado neste trabalho, mas os outros serão brevemente descritos a seguir:

O Logstash é responsável por receber os *logs*, fazer o *parse*, ou seja, transformar dados não estruturados em dados estruturados e envia-los para o banco de dados. Esta ferramenta é altamente configurável devido a sua arquitetura, ela é divida em 3 etapas: entrada(*input*), responsável por determinar os formatos e origens dos dados a serem inseridos; processamento(*filter*), cuja função é filtrar as informações através de regras; saída(*output*), responsável por determinar os destinos das informações processadas. Cada etapa possuí seus *plugins*, que são programas que integram com o *Logstash*. Através deles determina-se o fluxo de cada etapa, onde essas configurações devem ser salvas em um arquivo, que é passado como parâmetro na inicialização da ferramenta [CHNAJED 2015].

O Elasticsearch é um banco de dados não relacional e motor de buscas distribuído, desenvolvido para indexar documentos e fazer buscas baseadas em textos. Beats é uma

ferramenta extremamente leve, encarregada de fazer a transferência dos dados do servidor em que são gerados os *logs* (honeypot), para o Elasticsearch onde serão analisados; ou para uma instância do Logstash para que primeiramente sejam processados. Dentro da ferramenta proposta neste artigo, o Beats se encarregará de enviar os *logs* para serem processados, além de facilitar que sejam utilizados quantas aplicações *Honeypot* forem desejadas para alimentar o gerador de regras.

4. Proposta

Para chegar no objetivo proposto, será desenvolvido um sistema de prevenção contra ataques, o GARF, acrônimo para Gerador Automatizado de Regras em *Firewall*. Será composto por serviços *Open source*. Seu objetivo é analisar *logs* de acesso disponibilizados por servidores *Honeypot* e, com auxílio de uma solução de *Firewall*, bloquear de forma automática o acesso de possíveis atacantes aos servidores por período de tempo pré-determinado. Como o propósito de um *Honeypot* é possibilitar e facilitar invasões, todas as informações obtidas dele serão consideradas como possíveis ameaças. Assim será possível construir uma base sólida com informações sobres possíveis atacantes.

O *Honeypot* registra todas ações de um atanque em seus *logs*, possibilitando que capturemos essas informações. Para transferir estas informações ao GARF utilizamos o Beats. A função deste é coletar e enviar dados para uma instância do Logstash. Ele fica vigiando o arquivo de *log* e a cada mudança no arquivo, envia os registros para o Logstash em tempo real. Com isso nosso banco de dados estará sempre atualizado. Uma instância do Logstash pode receber dados de diversas instâncias do Beats, facilitando e possibilitando que nossa aplicação seja alimenta por mais de um *Honeypot*.

Os arquivos de *logs* coletados, podem ser provenientes de diversos tipos de Honeypot, ou seja, não necessariamente possuem uma estrutura homogênea. Este entrave é resolvido através do uso do Logstash para filtrar os arquivos e adquirir as informações realmente relevantes para a aplicação. Cada sistema Honeypot possuí a sua maneira de registrar os acesso, por isso pode ser necessário utilizarmos uma - ou mais - regras para cada estrutura de *log*. Levando as possíveis diferença estruturais dos *logs* em consideração, foi necessário configurar alguns *plugins* que executam as tarefas de entrada, filtragem e saída. Como as informações coletadas serão enviadas pelo Beats, então precisamos configurar o plugin do Beats para sabermos como receber os dados. Esse plugin permite definirmos os IPs e portas que o Logstash ficará escutando e esperando receber os dados enviados. Para fazer o *parsing* e filtragem utilizaremos o *plugin Grok* que permite a configuração de regras para processamento de dados não estruturados. Cada tipo de *log* possuíra sua regra de filtragem, além disso foi definido uma regra que faz com que apenas sejam inseridos - no banco de dados- as informações que combinem com os padrões definidos. Assim, podemos definir que as informações a serem buscadas são: IP de origem, porta de destino, data de acesso e o protocolo. Para definir a saída dos dados já filtrada será o utilizado o *plugin* Elasticsearch, que permite a inserção dessas informações em uma instância do Elasticsearch que indexará e disponibilizará as informações para consulta.

Foi escolhido o Elasticsearch ao invés de um banco de dados relacional devido a sua performance em buscas por texto e pela fácil integração com as ferramentas Logstash e Beats, facilitando e acelerando a disponibilidade das informações dos atacantes para o gerador de regras da aplicação que será descrito na sequência.

Com as informações a disposição, a solução poderá executar uma busca em todos

os dados que foram salvos desde sua última execução e recuperar aqueles considerados suspeitos e que serão bloqueados. Com essas informações em mãos, o módulo irá gerar e aplicar todas as regras no firewall. Além disso, haverá um módulo responsável por analisar e remover todas as regras que já tiveram seu período de duração expirados. As informações que forem suspeitas serão colocadas em um *Brown list*, que é uma lista contendo os IPs suspeitos, facilitando as futuras análises da ferramenta. Essa segunda funcionalidade será executada em períodos que houverem uma quantidade menor de dados a serem processados.

Adicional a análise dos *logs*, a ferramenta possuirá uma interface de configuração, que tem como propósito o ajuste de parâmetros como: tempo de duração das regras de bloqueio no *Firewall*, o intervalo de tempo entre as execuções do módulo de geração de regras e o número de ocorrências necessárias para que o atacante seja bloqueado. Através dessa interface também será possível remover regras individualmente. Estas informações serão salvas em um arquivo de configuração da aplicação. O fluxo dos dados e a interação entre as ferramentas utilizadas podem ser observados na Figura 2.

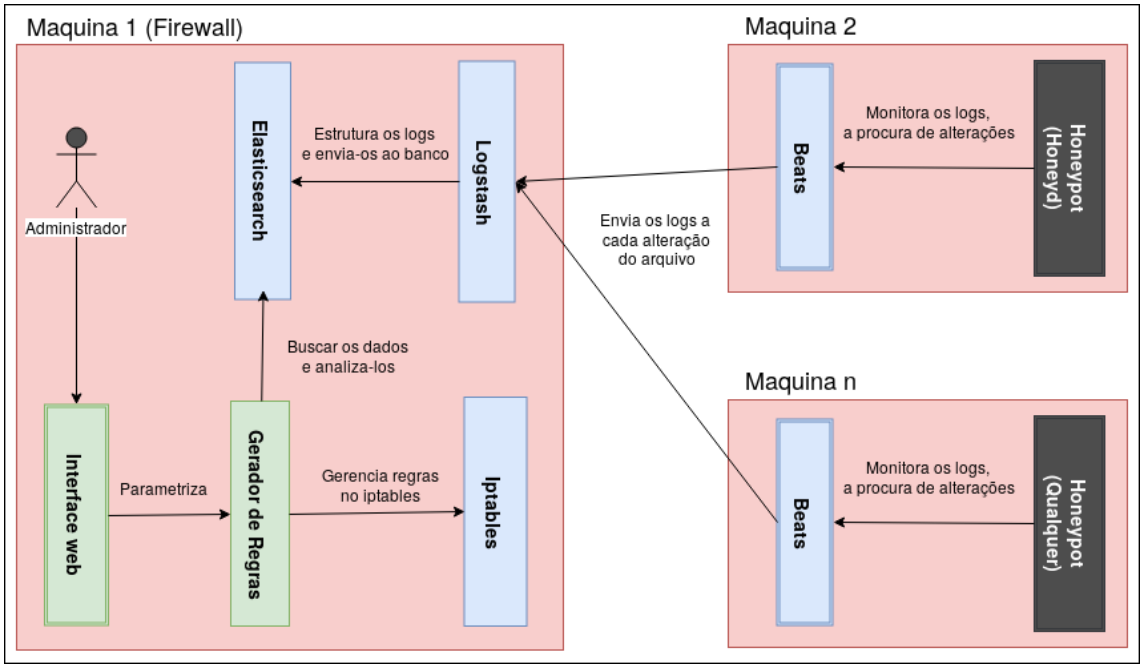

Figura 2. Diagrama de Visão Geral do sistema

5. Metodologia

Para que fosse possível o desenvolvimento de uma ferramenta de segurança preventiva, diversas etapas foram previamente definidas e executadas. Esta seção tem como objetivo descrever todos as etapas realizadas e o método de validação.

Primeiramente foi necessário buscar referências bibliográfica dos seguintes conceitos da área de administração de servidores: *Honeypot, Firewall, IPTables*, tipos de ataques e protocolo *Syslog*. Esta etapa foi necessária para que houvesse uma ambientação da área a ser abordada e uma visão mais clara do problema a ser resolvido no decorrer deste projeto. Em uma segunda fase, foi realizada uma pesquisa exploratória com

objetivo de analisar ferramentas que propõem processar e analisar *logs*. São elas: Pilha ELK, Splunk, Kippo, fail2ban e o sistema proposto no artigo escrito por Sunil Gupta (2012), *Logging and Monitoring to Detect Network Intrusions and Compliance Violations in the Environment*

Ao final destas duas etapas, ficou claro quais os problemas que as soluções que estão no mercado não solucionavam e qual o escopo da ferramenta a ser desenvolvida. Assim foi possível levantar quais funcionalidades a ferramenta teria. A principal tarefa seria analisar os *logs* de acessos de um *Honeypot* e gerar regras de bloqueio em um *Firewall* utilizando a *IPtables*, por ser amplamente utilizado em distribuições Linux.

Após definirmos as funcionalidades que seriam executadas, foram analisadas algumas tecnologias para serem utilizadas na construção da ferramenta. Entre elas estão: O *Logstash*, ferramenta de análise de *logs*; Os *Honeypots* Kippo e Honeyd foram analisados para que fosse determinado qual seria o escolhido para obter os *logs* a serem analisados. O primeiro é um *Honeypot* específico para o serviço de SSH e já o segundo - que foi o escolhido - é um sistema mais completo que possibilita que vários serviços e tipos de hosts sejam simulados;

Para conectar o servidor *Honeypot* com o *Firewall* foi utilizado o *Beats*, que tem como objetivo de estabelecer uma conexão entre a máquina que em que o *Honeypot* está sendo executado e a instância do *Logstash* para que os *logs* registrado por este seja enviado para aquele e o *IPtables*, para o gerenciamento das regras do *Firewall*.

A ferramenta foi desenvolvida usando a linguagem de programação python e utilizando o banco de dados não relacional *Elasticsearch*. Tendo as funcionalidades e as tecnologias definidas, foi elaborado a arquitetura da solução. Essa, está representada na Figura 2. Além do gerador de regras foi necessário elaborar uma interface web para que os usuários do sistema pudessem configurá-lo e acompanhar quais regras foram inseridas pela aplicação. Nesta parte do sistema foi utilizado o Flask, que é um micro framework python e o SpectreCSS que é um framework css para estilizar as páginas.

No processo de desenvolvimento foi utilizado a metodologia ágil *Scrum*. O software foi quebrado em diversas tarefas que ficaram armazenadas em um *backlog*. O critério escolhido para definir cada tarefa é que cada uma deveria ter um curto tempo de execução e deveria agregar algum valor ao produto final. Devido ao curto período de tempo determinado para o desenvolvimento e validação do projeto, cada *sprint* ocupou duas semanas. No primeiro dia de cada *sprint*, foi determinado, a partir do *backlog*, quais seriam as tarefas que seriam executadas naquela etapa. No fim de cada *sprint*, um pedaço funcional do produto foi entregue, revisado e testado - onde foi analisado o retorno das funcionalidades entregues e, se necessário, alterações no escopo do projeto foram realizadas alcançar o objetivo proposto. Todas alterações consequentes da etapa anterior foram adicionadas ao backlog para que fossem executadas nas etapas seguintes. Foi escolhida essa metodologia devido as entregas incrementais propostas por ela, isso permitiu validar desde o começo o produto e alterá-lo caso seja necessário, evitando possíveis atrasos ou a entrega de um produto que não cumpra o que foi planejado.

6. Validação

Para validar a solução proposta, foi utilizado um ambiente de testes local composto por duas máquinas virtuais. Cada uma desempenhou a função de uma entidade envolvida. A máquina 1 foi instalado o *Honeypot* Honeyd para coletar os *logs* de acesso e o Beats para

monitorar e enviar os dados para a segunda máquina. Nesta foram instalados o Logstash para receber os dados e analisá-los, o Elasticsearch para armazenar as informações processadas e o GARF para gerar as regras no firewall.

A máquina 1 foi populada com algumas semanas de *logs* reais que foram registrados por uma instância do Honeyd de propriedade do CERT-RS [3]. O programa foi executado durante 10 dias e ao final deste período, foram geradas 904 regras. A partir de uma analise destas foi possível verificar que entre os 5 IPs com maior número de regras geradas, 4 são oriundos da Europa e um da China. Além disso os serviços com maior número de regras geradas foram o SMB, que roda na porta 445 com um total de 665 regras e o SSH que roda na porta 22 com um total de 107 regras. O primeiro (SMB), é um protocolo de compartilhamento de arquivos presente em servidores Windows que já foi explorado por *worms* como o Sasser e Nimda e o segundo (SSH) é um protocolo de acesso remoto ao shell.

Além das regras geradas com o objetivo de proteger o servidor temporariamente, a aplicação forneceu dados que podem ajudar os administradores a tomarem medidas mais prolongadas para proteger os dados presentes neste. Com isso, é possível afirmar que a aplicação desenvolvida cumpriu o que foi proposto.

7. Considerações finais

Através dos estudos realizados, foi possível evidenciar uma escassez de soluções que trabalhem com prevenção contra ataques. Durante o desenvolvimento deste trabalho, foi apresentado a tecnologia do *Honeypot*, como sua funcionalidade de coletar dados sobre atacantes pode ser um recurso importante para a área de segurança da informação e como é pouco utiliza por aplicações. A partir disto, foi proposto e desenvolvido um sistema cujo objetivo principal seria utilizar esses dados coletados por sistemas *Honeypots* para gerar regras em *firewall*.

Uma das contribuições do projeto, desde o começo, era colaborar com a área de segurança da informação através da utilização de dados fornecidos por Honeypots para elaborar uma ferramenta de segurança, destinada a servidores, que pudesse ser utilizada de maneira preventiva para proteger estes de serem atacados. Após o fim do desenvolvimento a aplicação foi submetida a um período de validação de 10 dias, onde 904 regras de firewall foram geradas no servidor de teste. A partir destas regras foi realizada uma análise para discriminar quais foram os IPs mais presentes nas regras geradas e quais foram os serviços mais procurados para os ataques. Dados estes que podem auxiliar nas tomadas de decisões dos administradores de servidores, comprovando o funcionamento do *software* desenvolvido.

Para que a aplicação fosse desenvolvida e validada dentro do período prédeterminado para o projeto, foi necessário limitar o escopo desta. Com isso, algumas funcionalidades descobertas durante do desenvolvimento do trabalho proposto ficarão como sugestões para trabalhos futuros. A primeira, seria a expansão do gerador para o protocolo IPv6. Para desenvolver esta funcionalidade seria necessário adicionar uma regra no Logstash para fazer *parsing* o IPv6 de origem dos *logs*. Além disso, como as regras do Firewall para o IPv6 são gerenciadas pela aplicação IP6tables, seria necessário adaptar os métodos que interagem com as regras para utilizarem esta ferramenta. Uma segunda funcionalidade que agregaria no projeto, seria a adição de um mapa que demarcasse a

[3]Grupo de resposta a incidentes de segurança para a Rede Acadêmica Gaúcha [CERT-RS 2018]

posição geográfica dos IPs que foram bloqueados pelo GARF. Com isso, o administrador conseguiria visualizar a origem dos acessos de uma maneira mais prática.

Referências

CARASSO, D. (2012). *Exploring Splunk*. CITO Research, 1th edition.

CERT-RS (2018). Computer emergency team - rio grande do sul.

CHNAJED, S. (2015). *Learning ELK Stack*. Packt Publishing, 1th edition.

DEVERIYA, A. (2005). *Network Administrators Survival Guide*. Cisco Press, 1th edition.

ELLINGWOOD, J. (2014). How to protect ssh with fail2ban on ubuntu 14.04.

ELLINGWOOD, J. (2015). Honey pot systems explained.

EVEN, L. R. (2000). Honeypot systems explained.

GROSS, G. (2015). Botnet detection and removal: Methods & best practices.

GUPTA, S. (2012). Logging and monitoring to detect network intrusions and compliance violations in the environment. *The SANS Intitute*.

KIZZA, J. M. (2015). *Guide to Computer Network Security*. Springer, 3th edition.

NETO, U. (2004). *Dominando Linux Firewall IPtables*. Ciência Moderna, 1th edition.

OSBOURNE, C. (2017). The average ddos attack cost for businesses rises to over $2.5 million.

PAYÃO, F. (2016). Quebrando a internet: estamos sofrendo o maior ataque ddos da história.

RAWAT, M. (2014). Tracking attackers with a honeypot - part 2.

SPITZNER, L. (2002). *HoneyPtos: Tracking Hackers*. Addison-Wesley Longman Publishing, 1th edition.

STALLINGS, W. (2011). *Cryptography and Network Security: Principles and Pactice*. Addison-Wesley Longman Publishing, 5th edition.

TCU (2012). *Boas Práticas em Segurança da informação*. Tribunal de Contas da União, 4th edition.

VIDAL, W. R. C. (2015). Estudo de viabilidade para detecção de intrusão usando técnicas de big data para a análise de logs. *UFRGS*.