Quebrando a Barreira entre Mecanismos de Segurança através da Composição de Serviços *Web*: Uma Arquitetura para Detecção de Ataques Distribuídos e de Múltiplas Etapas

Leonardo Lemes Fagundes[1], Luciano Paschoal Gaspary[2]

[1]Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos
[2]Instituto de Informática
Universidade Federal do Rio Grande do Sul
llemes@unisinos.br, paschoal@inf.ufrgs.br

Abstract. *In the recent years , the number of planned attacks such as DDoS, has increased. These attacks are composed of several stages and depart from a number of hosts. Traditional intrusion detection solutions do not cope well with this type of attack because, among other reasons, they lack mechanisms for uniform communication with distinct security systems (e.g. IDS, firewall, etc.) and to correlate, in a timely manner, the events observed. To fulfill the mentioned gap, this paper proposes a service oriented architecture for multi-stage, distributed attack detection. The architecture has been developed following the WSDM (Web Services Distributed Management) standard and evaluated experimentally using a DDoS attack scenario proposed by the MIT Lincoln Laboratory.*

Resumo. *Nos últimos anos tem se observado o avanço de ataques planejados, como DDoS, compostos de múltiplas etapas e partindo de diversos hosts. As soluções mais tradicionais para detecção de intrusão não estão preparadas para lidar com essa natureza de ataque devido, entre outros fatores, à carência de mecanismos para comunicação uniforme com diferentes sistemas de segurança (ex: IDS, firewall, etc.) e para correlação, em tempo hábil, dos eventos observados. Para suprir tal lacuna, este artigo propõe uma arquitetura orientada a serviços para detecção de ataques distribuídos e de múltiplas etapas. A arquitetura foi desenvolvida com base no padrão WSDM (Web Services Distributed Management) e avaliada experimentalmente usando cenário de ataque DDoS proposto pelo MIT Lincoln Laboratory.*

1. Introdução

Nos últimos anos tem se observado o avanço de ataques de *múltiplas etapas*, partindo de diversos *hosts*, sendo realizado contra organizações. Embora não haja consenso na literatura, esses ataques podem ser expressos por um conjunto de atividades maliciosas, executadas de forma distribuída, ao longo de um intervalo de tempo [Gaspary et. al 2005]. Essas etapas ou *ataques intermediários* possuem objetivos específicos e, em alguns casos, podem ser realizadas de diferentes maneiras e em uma ordem temporal qualquer [Cheung et al., 2003].

Um exemplo simplificado dessa natureza de ataque pode ser ilustrado por um cenário em que um atacante deseja expor informações confidenciais que deveriam ser acessadas somente por usuários autorizados, a partir de um servidor *web*. Inicialmente esse atacante poderia identificar uma vulnerabilidade, por exemplo um *buffer overflow*, na implementação do módulo SSL (*Secure Socket Layer*) desse servidor. A partir da exploração dessa vulnerabilidade, o atacante obteria privilégios que lhe permitiriam executar comandos remotamente. Então, acessaria o sistema de arquivos, localizaria os dados que desejasse obter e alteraria a página *web* inicial desse servidor. Com isso, através de uma requisição HTTP realizada a partir de qualquer estação, seria possível acessar esses dados. Tal ataque pode ser nitidamente caracterizado pela execução de um conjunto de passos, que são potencialmente registrados por diferentes mecanismos de segurança como IDSs, *Host Intrusion Detection Systems* (HIDs), arquivos de *log* vinculados ao processo de autenticação e ao serviço *web*. Ataques de negação de serviço distribuídos – como *Tribe Flood Network* (TFN) [Dittrich, 1999] e *Mstream* [Dittrich, 2000] – *worms* também são exemplos de intrusões baseadas em múltiplas etapas.

As estratégias ou planos de intrusões são compostos a partir do desenvolvimento de um conjunto de etapas que pode ocorrer de forma seqüencial ou concorrente. Essas etapas, quando devidamente identificadas, podem conduzir não só à detecção das atividades de intrusão, ainda nas etapas iniciais, mas também à interrupção dessas ações [Northcutt, 2000]. Portanto, o desafio é identificar rapidamente a realização dessas etapas e prevenir que o intruso realize os passos subseqüentes.

Grande parte das soluções de detecção de intrusão propostas até o momento se restringe a detectar atividades muito pontuais, não identificando o vínculo existente entre atividades relacionadas. Sendo assim, fica a cargo dos administradores auditar o ambiente atingido em busca de evidências desses ataques, a fim de entender a estratégia existente por traz dos mesmos e, então, executar procedimentos preventivos no intuito de evitar novos ataques dessa mesma natureza.

Trabalhos de pesquisa recentes com o objetivo de projetar e desenvolver arquiteturas ou *frameworks* para detecção de intrusão contemplam mecanismos para agregação de eventos[1] [Debar et al., 2001; Cuppens et al., 2002; Ning et al., 2002; Porras et al., 2002]. Embora permitam reunir eventos gerados em diferentes pontos de uma rede, o seu processamento e a sua análise são atrasados, uma vez que os algoritmos envolvidos são executados, por exemplo, sobre o conjunto de eventos coletados em um turno ou mesmo em um dia de operação (análise *post mortem*). Tal procedimento implica um ciclo longo e, portanto, pouco apropriado para detectar ataques em fase inicial.

Outra limitação relevante no que se refere à detecção de ataques de múltiplas etapas reside na carência de mecanismos apropriados para representar os cenários de intrusão. Embora algumas soluções proponham notações, através das quais é possível definir as atividades que compõem um ataque, elas são restritas em relação (*i*) à

[1] O termo *evento* é adotado em todo o artigo para expressar informações (ex: entradas em arquivos de *log* e pacotes capturados) geradas por mecanismos de segurança como IDSs, HIDs, entre outros. Já o termo *alerta* é utilizado para representar mensagens, de mais alto nível, que são enviadas ao gerente de segurança para informá-lo, por exemplo, sobre a ocorrência de um ataque.

natureza dos eventos que podem ser representados, (*ii*) às possibilidades de ordenamento dos eventos a serem observados e (*iii*) às possibilidades de avaliação de atributos desses eventos.

Por fim, representa uma limitação *considerável* nas soluções propostas a falta de uma maneira uniforme e bem definida para interagir com diferentes mecanismos de segurança visando à obtenção de eventos (e sua correlação) e à solicitação de execução de procedimentos (ex: reconfiguração ou reinício). O mais longe a que se conseguiu chegar nessa direção foi a definição de um padrão pelo IETF – o IDMEF (*Intrusion Detection Message Exchange Format*) – que permite que diferentes tipos de sensores reportem a ocorrência de eventos usando uma linguagem única. Ao adotar essa linguagem, sistemas de detecção de intrusão como Snort [2006] geram eventos "padronizados" e os enviam a uma entidade central para processamento e análise (ex: Prelude [2006]), recaindo na primeira limitação mencionada.

Para suprir a lacuna recém apresentada, este artigo propõe uma arquitetura para detecção de ataques distribuídos e de múltiplas etapas baseada na composição de serviços *web* voltados à segurança. Graças a características nativas de serviços *web* – como a provisão de mecanismo uniforme de comunicação e a capacidade de esconder aspectos de implementação dos serviços (cingindo sua heterogeneidade) – a arquitetura prevê a utilização de um serviço a frente de cada mecanismo de segurança (ex: *firewall*, NIDS, HIDS, etc.). Estes serviços fazem a interlocução na comunicação entre a aplicação de gerenciamento e os mecanismos propriamente ditos, permitindo a solicitação seletiva de eventos observados, bem como a invocação de ações de contenção. Com o apoio de uma linguagem, também proposta neste artigo, o gerente de segurança especifica cenários de intrusão de múltiplas etapas e, por intermédio da aplicação de gerenciamento, instancia sua monitoração *assinando* por eventos junto aos serviços envolvidos no cenário e delegando a observação do cenário a um serviço de detecção (que receberá os eventos que foram subscritos). À medida em que um determinado cenário de ataque evolui e vai sendo gradativamente detectado, *on-the-fly*, medidas de contenção podem ser invocadas.

Em consonância com as tecnologias atuais de gerenciamento, a arquitetura foi desenvolvida com base no padrão WSDM (*Web Services Distributed Management*) [Vambenepe et al. 2005], padronizado pelo OASIS em maio de 2005. É importante destacar que, até onde sabemos, este é o primeiro trabalho a propor o emprego do *framework* WSDM na proposta de uma solução para gerenciamento de segurança.

O artigo dá continuidade às idéias lançadas em [Fagundes et al. 2005], publicado com artigo curto no SBSeg 2005, desdobrando os conceitos seminais lá apresentados em (*i*) uma linguagem para representação de cenários de intrusão e (*ii*) uma arquitetura que foi desenvolvida e avaliada. O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta uma síntese dos trabalhos relacionados. A Seção 3 introduz a linguagem e a arquitetura propostas para detecção de cenários de ataques baseados em múltiplas etapas. Os resultados obtidos a partir de avaliação experimental da arquitetura são descritos na Seção 4. A Seção 5 encerra o artigo com considerações finais e perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

A agregação e a correlação de eventos têm sido abordadas em diversos trabalhos vinculados à detecção de intrusão [Debar et al., 2001; Cuppens et al., 2002; Porras et al., 2002]. A principal limitação identificada nesses trabalhos reside na necessidade de coletar, previamente, todos os eventos a serem analisados para que *sobre eles* possam ser aplicadas técnicas da Inteligência Artificial. Com isso, a deteção de potenciais ataques só ocorre muito após a sua ocorrência (análise *post mortem*).

Ning em [2002] apresenta um método que correlaciona pré-requisitos e conseqüências de eventos gerados por sistemas de detecção de intrusão a fim de determinar os vários estágios de um ataque. Os autores sustentam o argumento de que um ataque geralmente tem diferentes estágios e não acontece isoladamente, ou seja, cada estágio do ataque é pré-requisito para o próximo. Por exemplo, uma varredura de portas pode identificar os *hosts* que possuem serviços vulneráveis; com base nisso, o atacante pode explorar esses *hosts* para executar código arbitrário com privilégios do sistema local ou causar uma negação de serviço. O fato de pré-requisitos e conseqüências serem modelados como *predicados* dificulta a implantação em larga escala da abordagem, uma vez que a definição desses predicados não é uma tarefa fácil e a base de casos precisa ser constantemente atualizada, o que requer trabalho substancial. A proposta também é limitada ao não ser efetiva para identificar ataques onde a relação causa e conseqüência não pode ser estabelecida. Por exemplo, dois ataques (*Smurf* e SYN *flooding*) disparados quase ao mesmo tempo contra o mesmo alvo a partir de dois locais diferentes não seriam relacionados (embora exista forte conexão entre eles: mesmo instante e mesmo alvo).

Language to Model a Database for Detection of Attacks (LAMBDA) [Cuppens, 2000], *State Transition Analysis Technique Language* (STATL) [Eckmann, 2002] e *Correlated Attack Modeling Language* (CAML) [Cheung et al., 2003] representam um sub-conjunto representativo de linguagens para modelagem de ataques. Essas não são capazes de representar eventos oriundos de quaisquer mecanismos de segurança (ex: NIDS e HIDS). Ademais, possuem *expressividade* limitada no que se refere à avaliação de atributos e à especificação de *ordem* de eventos (ex: seqüência, paralelismo e ciclos), dificultando a especificação de determinados cenários.

Em relação a propostas para comunicação uniforme com diferentes sistemas de segurança – conforme mencionado na Introdução – o principal avanço alcançado até o momento foi a criação do *formato padrão* IDMEF. Em síntese, esse padrão não regulamenta interações com mecanismos de segurança, o que permitiria, por exemplo, (*i*) a solicitação por um conjunto selecionado de eventos (através de esquemas *publish/subscribe* ou *request/response*) para análise remota, (*ii*) a solicitação por atualização de assinaturas (no caso de um sistema antivírus) e (*iii*) a reconfiguração dos mecanismos de segurança propriamente ditos (como medida de contenção).

Prelude [2006] é um sistema de detecção de intrusão que adota o padrão IDMEF, sendo capaz de receber eventos gerados por diferentes sensores. Esses eventos, uma vez consolidados em uma estação central, dão origem a um conjunto de estatísticas na forma de relatórios e gráficos. Tais informações não são suficientes para determinar ataques distribuídos e de múltiplas etapas que estejam em curso. Além disso, o sistema não oferece mecanismos para executar medidas de contenção.

3. A Arquitetura SecCompose

SecCompose consiste de uma linguagem para a modelagem de ataques distribuídos e de múltiplas etapas e de uma arquitetura de *software* orientada a serviços para detectá-los. As Seções 3.1 e 3.2 abordam, separadamente, esses componentes.

3.1 Representação de Cenários de Intrusão

Considerando as limitações das linguagens existentes para representar ataques de múltiplas etapas, discutidas na seção anterior, propomos uma linguagem alternativa denominada *Multistage Attack Description Language* (MADL). A linguagem possui duas notações: uma gráfica (G-MADL) e outra textual (T-MADL). A primeira consiste de uma adaptação dos diagramas de atividades da *Unified Modeling Language* (UML), ao passo que a segunda é uma notação baseada em XML. Essas notações não são equivalentes, pois a notação textual permite uma descrição mais detalhada das atividades que fazem parte de um cenário de ataque. Já a notação gráfica, por sua vez, equivale a um sub-conjunto da notação textual, e oferece a possibilidade de representar visualmente os cenários de ataque em um alto grau de abstração. A seguir são descritas as principais características de ambas as notações. Mais detalhes sobre as mesmas podem ser obtidas em [Fagundes, 2006].

3.1.1 G-MADL

Os elementos básicos da especificação visual de um cenário de ataque são representados na Figura 1. Quatro símbolos distintos representam (1) início do cenário, (2) atividades, (3) seqüência entre atividades e (4) término do cenário. Uma atividade pode representar um evento a ser observado ou uma ação a ser executada, tal como o envio de um alerta ao administrador ou a invocação de um procedimento de contenção. Os rótulos dentro de uma atividade, assim, identificam, respectivamente, o evento (ou o procedimento) e o serviço de segurança que deverá gerá-lo (ou executar o procedimento de contenção). No exemplo está sendo representado que o evento *BackdoorRst.bScan* deve ser observado antes do *Rst.bShellCommand*, ambos oriundos do serviço IDS 1.

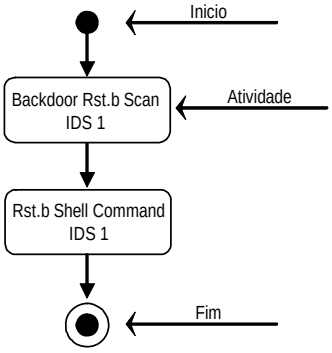

Figura 1 - Elementos básicos de G-MADL

Em relação às possibilidades de ordenação de atividades, G-MADL oferece construtores para representação de *paralelismo* e *condicionalidade*. O primeiro permite representar a observação de eventos e/ou execução de ações que devem ocorrer em paralelo, sem uma ordem determinada. Este é o caso do exemplo apresentado na Figura 2(a), em que é especificado o interesse de observar a instalação do software *Mstream* em três estações, independente da ordem. O segundo construtor, por sua vez, permite especificar situações em que a observação de uma atividade, dentre um conjunto, é suficiente para que o ataque passe para o estágio seguinte. Por exemplo, na Figura 2(b) a ocorrência de uma das três atividades inclusas no construtor de condicionalidade faz com que o cenário de ataque evolua para a atividade seguinte, *TFTP Connection*.

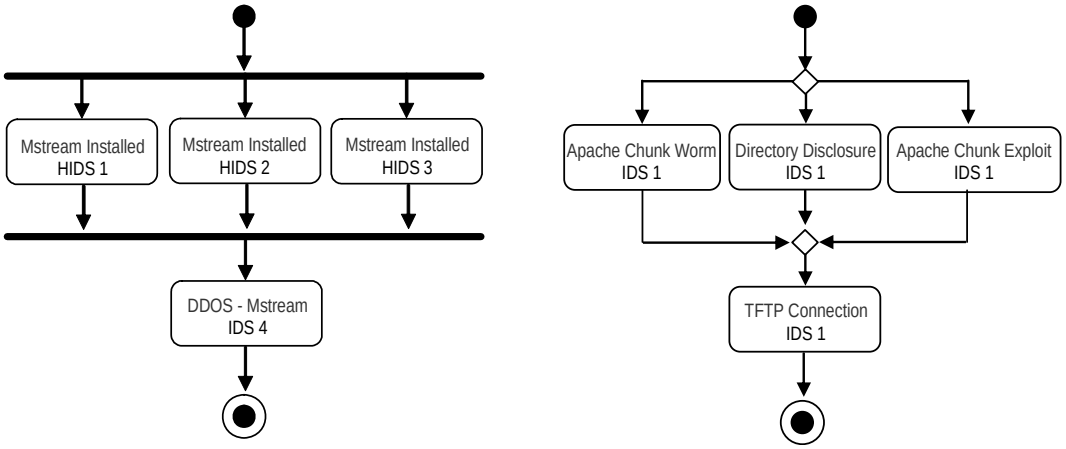

Figura 2 - Representação de paralelismo e condicionalidade

3.1.2 T-MADL

Baseada na linguagem XML, T-MADL permite detalhar as atividades que compõem um cenário de ataque, além de meramente ordená-las no tempo. A Figura 3 ilustra a especificação textual do cenário apresentado anteriormente na Figura 1. A descrição inicia com <scenario> e encerra com </scenario> (linhas 1 e 6). Em seguida é empregado o construtor <sequence> e </sequence>, envolvendo as atividades que devem ocorrer na seqüência (linhas 2 e 5). Por fim, são enumeradas as atividades, no caso a observação de dois eventos (tag event): *Backdoor Rst.b Scan* e *Rst.b Shell Command* (linhas 3 e 4).

```
1 <scenario>
2     <sequence>
3         <event name="Backdoor Rst.b Scan" ip_dst="192.168.10.5"/>
4         <event name="Rst.b Shell Command" ip_dst="192.168.10.5"/>
5     </sequence>
6 </scenario>
```

Figura 3 - Representação textual de um cenário de ataque

Para expressar paralelismo e condicionalidade, as *tags* a serem utilizadas são <parallel> e <choice>, respectivamente. Esses construtores, junto com o <sequence>, podem ser combinados e aninhados, de modo a viabilizar a representação de cenários de ataque complexos (como o ilustrado na Seção 4). Já as atividades previstas na linguagem são <event>, <action> e <alert>. Cada uma delas possui atributos específicos, enumerados a seguir:

- *Event:* nome do evento a ser observado, endereços IP da origem do mesmo e da estação alvo, portas local e remota, data e hora de ocorrência, entre outros;
- *Action:* endereço do serviço *web* a ser invocado e parâmetros;
- *Alert:* descrição da mensagem a ser enviada ao gerente de segurança.

Caracteres especiais como *asterisco*, *interrogação* e *exclamação* podem ser utilizados na definição de alguns atributos. O asterisco é empregado para substituir uma

cadeia qualquer de caracteres. Já os caracteres interrogação e exclamação representam a substituição de um único caracter e a negação de uma cadeia de caracteres.

3.2 Visão Conceitual da Arquitetura

A arquitetura SecCompose é organizada em quatros componentes: aplicação de gerenciamento, serviços de notificação, serviço de detecção e serviços de contenção.

- *Aplicação de gerenciamento*: componente através do qual o gerente de segurança (*i*) especifica os cenários de intrusão, (*ii*) instancia o processo de monitoração e (*iii*) visualiza os alertas gerados pelo serviço de detecção.
- *Serviços de notificação*: são responsáveis por observar e comunicar a ocorrência de eventos. As principais funções dos serviços de notificação são: (*i*) publicar nos repositórios de serviços *web* as suas funcionalidades, (*ii*) receber subscrições por eventos e (*iii*) notificar o serviço de detecção à medida que os eventos subscritos forem sendo observados.
- *Serviço de detecção*: é responsável pela monitoração das especificações de cenários de ataques. As principais funções exercidas por esse serviço são: (*i*) receber as mensagens de notificação , (*ii*) avaliar a evolução de ataques, (*iii*) gerar alertas e (*iv*) invocar os serviços de contenção.
- *Serviços de contenção*: são responsáveis pela execução de procedimentos cujo objetivo é inibir a evolução de um ataque.

A Figura 4 ilustra uma visão geral da arquitetura, incluindo esses componentes e as interações previstas entre eles. Inicialmente, os serviços de notificação e contenção são publicizados (fluxos 1 e 2 da figura) em um repositório de serviços *web*. Um serviço de notificação para o IDS Snort, por exemplo, oferece suporte à assinatura para todos os possíveis eventos gerados por ele (organizados na forma de *propriedades*). As possibilidades de assinatura, nesse caso, ficam acessíveis no repositório para consulta. Já um serviço de contenção associado a um determinado *firewall*, que provê suporte à reconfiguração de regras de filtragem, divulgará no repositório o método a ser invocado para tal e os parâmetros a serem informados.

Ao acessar a aplicação de gerenciamento (3), o gerente de segurança pode realizar consultas ao repositório de serviços (4) com o objetivo de obter uma lista dos serviços de notificação e contenção disponíveis. As informações fornecidas por esses serviços são utilizadas na modelagem dos cenários de intrusão que, uma vez especificados, são armazenados em uma base de dados local (5). O passo seguinte nessa arquitetura ocorre quando o gerente decide passar a monitorar a ocorrência de um determinado cenário. Nesse momento, a aplicação de gerenciamento interage com os serviços de notificação envolvidos, subscrevendo pelos eventos de interesse (6), e com o serviço de detecção, repassando a ele a especificação do cenário a ser monitorado (7).

Assim que um dado serviço de notificação recebe subscrições oriundas da aplicação de gerenciamento, inicia o processo de monitoração dos eventos de interesse *assinados*. Tão logo seja identificada a ocorrência de um desses eventos (8), o serviço de detecção é comunicado (9). Para cada evento recebido, o serviço de detecção verifica se tal evento permite a evolução no fluxo previsto para algum dos ataques sendo monitorados. Além disso, dependendo do cenário, essa verificação pode resultar tanto

na invocação de serviço de contenção (10) quanto no envio de alerta à aplicação de gerenciamento (11).

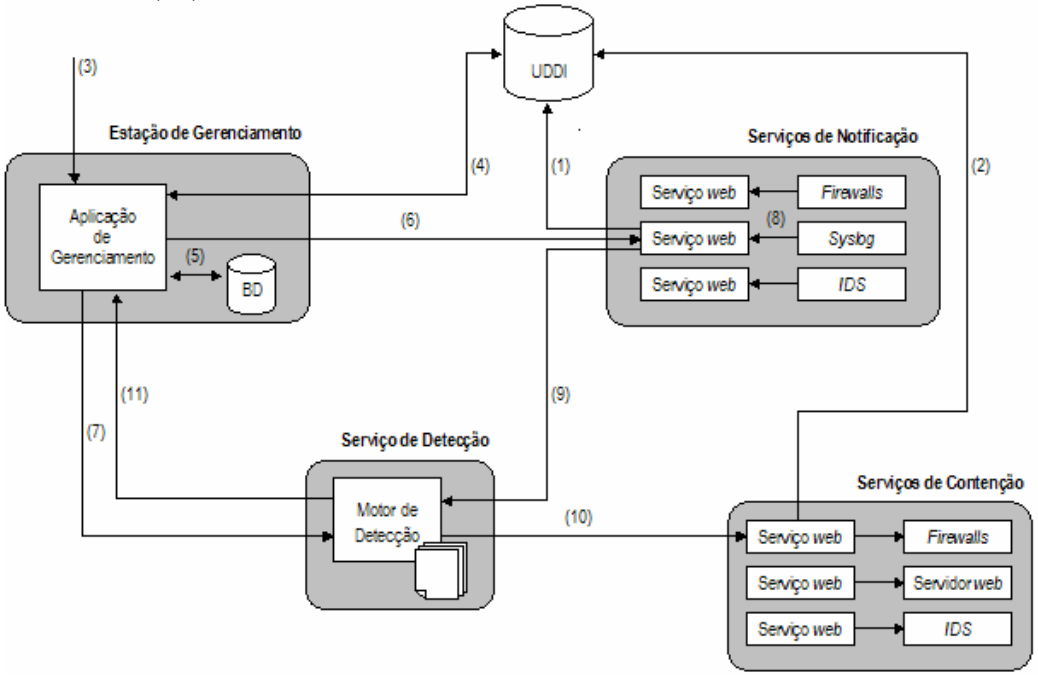

Figura 4 - Componentes de SecCompose e suas interações

3.3 Implementação

A implementação de SecCompose foi realizada com base no padrão WSDM [Vambenepe et al. 2005], fazendo uso ostensivo da sub-especificação *Web Services Base Notification* (WS-Base Notification) [Graham et al., 2004]. A autenticação e a integridade das mensagens trocadas entre os componentes da arquitetura são obtidas com o emprego do *Web Services Security* (WS-Security), padrão de segurança consolidado para serviços *web*.

A aplicação de gerenciamento constitui-se de um conjunto de *scripts* que, na atual implementação do protótipo, oferece suporte à definição de especificações em T-MADL, à organização do processo de instanciação dos cenários de ataque e à visualização dos alertas gerados pelo serviço de detecção.

Cenários de ataque especificados em T-MADL são verificados (sintaticamente) através da ferramenta XML Starlet Command Line XML Tollkit. A instanciação desses cenários, por sua vez, é realizada a partir de *scripts Bash* como o ilustrado na Figura 4. Nas linha 2-3 são realizadas subscrições ao serviço ws-snort (serviço de notificação associado ao Snort), pelos eventos (ou propriedades) *BackdoorRst.bScan* e *Rst.bShellCommand*. Como pode ser observado, esse serviço está acessível no endereço http://10.16.166.15:8080/pubscribe/services/arq/ws-snort, e o conteúdo (em XML) das respectivas solicitações de subscrição são lidas dos arquivos subscribe_BackdoorRst.bScan.soap e subscribe_Rst.bShellCommand.soap. Na linha 4 o serviço de detecção é invocado. Os parâmetros informados são dois: start, indicando que se deseja iniciar a monitoração de um novo cenário, e http://10.16.166.1/Rst.bShellCommand.madl, o endereço onde a especificação do cenário deve ser buscada.

```
1  ...
2  ant –Durl=http://10.16.166.15:8080/pubscribe/services/arq/ws-snort -Dxml= requests/subscribe_BackdoorRst.bScan.soap
3  ant –Durl=http://10.16.166.15:8080/pubscribe/services/arq/ws-snort -Dxml= requests/subscribe_Rst.bShellCommand.soap
4  http://10.16.166.50:8080/ws/service?type=ws-detection&para1=start&para2= http://10.16.166.1/Rst.bShellCommand.madl
5  ...
```

Figura 4 - *Script* para instanciação de um cenário de ataque

A subscrição por eventos junto aos serviços de notificação é realizada respeitando a especificação WS-Base Notification, através da primitiva Subscribe. A Figura 5 ilustra o formato de uma subscrição pelo evento *BackdoorRst.bScan*. Na linha 3 é informado ao serviço de notificação que os eventos (deste tipo) observados devem ser enviados ao endereço http://10.16.172.77:8081, local onde se encontra o serviço de detecção. Já na linha 7 é informado o nome do evento em si (no caso, *BackdoorRst.bScan*).

```
1  <wsnt:Subscribe>
2     <wsnt:ConsumerReference>
3        <wsa:Address>http:/10.16.172.77:8081</wsa:Address>
4        <wsa:ReferenceProperties/>
5     </wsnt:ConsumerReference>
6     <wsnt:TopicExpressionDialect="http://docs.oasis-open.org/wsn/2004/ 06/TopicExpression/Simple">
7        Fs: BackdoorRst.bScan
8     </wsnt:TopicExpression>
9  </wsnt:Subscribe>
```

Figura 5 - Formato de subscrição para o evento *BackdoorRst.bScan*

Os serviços de notificação foram implementados usando como base o *Pubscribe* [2006], uma implementação em Java da especificação WS-Base Notification que faz parte do projeto Web Services Project @ Apache. É importante destacar que cada tipo de evento gerado pelo mecanismo de segurança é representado no serviço de notificação por uma Property (mapeamento 1:1). Por exemplo, os eventos *BackdoorRst.bScan* e *Rst.bShellCommand*, gerados pelo Snort, são mapeados no serviço para *propriedades* de mesmo nome.

```
<Body>
   ...
   xmlns:fs=\"http://ws.apache.org/resource/example/arq\">
   <wsrp:Update>
      <fs: BackdoorRst.bScan >
      <fs:IP_src> 192.168.20.17 </fs:IP_src>
      <fs:IP_dst> 192.168.10.5 </fs:IP_dst>
      <fs:Date> 08/01/2006 </fs:Date>
      <fs:Time> 16:30 </fs:Time>
      </fs: BackdoorRst.bScan >
   </wsrp:Update>
   ...
</Body>
```

Figura 6 - Formato de mensagem para atualização do serviço de notificação

Junto aos serviços de notificação são executados *scripts* que – a cada nova entrada nos arquivos de *log* gerados pelo mecanismo de segurança em questão – extraem as informações disponíveis e atualizam o serviço (fluxo 8 na Figura 4), executando a primitiva SetResourceProperties para a propriedade correspondente. A Figura

6 ilustra os parâmetros informados na atualização da propriedade *BackdoorRst.bScan*, estruturados em XML (identificação, endereços IP origem e destino e data/hora).

. Sempre que uma requisição de subscrição é recebida, o serviço de notificação passa a monitorar a propriedade indicada. No momento que a mesma sofre atualização, o serviço gera uma mensagem de notificação e a envia ao serviço de detecção informado na subscrição. Na notificação ilustrada na Figura 7, é possível observar o valor atualizado (NewValue) para cada uma das informações (linhas 8-11) fornecidas para a propriedade em questão (linha 5).

```
1  <wsn:Message>
2      <wsrf:ResourcePropertyValueChangeNotification>
3          xmlns:wsrf="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.xsd">
4          <wsrf:NewValue>
5          <fs: BackdoorRst.bScan xmlns:wsrp="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-
6                ResourceProperties-1.2-draft-01.xsd" xmlns="http://schemas.xmlsoap.org/soap/envelope/"
7                xmlns:fs="http://ws.apache.org/resource/arq">
8              <fil:IP_src> 192.168.20.12 </fil:IP_src>
9              <fil:IP_dst> 192.168.10.5 </fil:IP_dst>
10             <fil:Date> 08/01/2006 </fil:Date>
11             <fil:Time> 16:45 </fil:Time>
12             </fs: BackdoorRst.bScan >
13         </wsrf:NewValue>
14     </wsrf:ResourcePropertyValueChangeNotification>
15 </wsn:Message>
```

Figura 7 - Formato de notificação para a propriedade *BackdoorRst.bScan*

O serviço de detecção foi implementado na forma de um serviço *web*, seguindo a especificação WSDM. Oferece suporte à solicitação pela monitoração e interrupção de cenários de ataque. Uma vez que o serviço tenha sido programado para monitorar um cenário, o mesmo é carregado em memória. A cada nova mensagem de notificação recebida, é realizada uma verificação que determina se o evento notificado permite a evolução de algum dos cenários monitorados.

Caso esteja previsto na especificação do cenário de ataque que a observação de um determinado evento deve resultar na geração de um alerta, o motor de detecção envia uma mensagem de notificação à aplicação de gerenciamento. Além disso, a ocorrência de um evento também pode provocar a invocação de um serviço de contenção, um serviço *web* que pode ser desenvolvido em qualquer linguagem de programação. O protótipo atual não contemplou a implementação de serviços de contenção.

4. Avaliação Experimental

Em um primeiro esforço para demonstrar que a arquitetura SecCompose é adequada para detectar ataques distribuídos e de múltiplas etapas, modelou-se cenário de ataque DDoS proposto pelo MIT Lincoln Laboratory, denominado LLDOS [DARPA, 2000]. O ataque é organizado em cinco etapas:

- *Sondagem de hosts ativos*: o objetivo desta etapa é identificar os *hosts* que estão ativos na rede;
- *Sondagem de hosts executando sadmind*: os *hosts* descobertos na etapa anterior são alvo de uma nova sondagem de portas, agora com o objetivo de identificar quais

desses *hosts* estão executando a ferramenta de administração remota denominada *sadmind*;

- *Exploração de vulnerabilidade do serviço sadmind*: nesta etapa o atacante executa o *exploit sadmind Remote-to-Root* com diferentes parâmetros contra cada um dos *hosts* identificados na etapa 2. O objetivo é executar remotamente comandos com privilégios do usuário *root*. A etapa encerra com a criação de uma nova conta de usuário em cada um dos *hosts;*

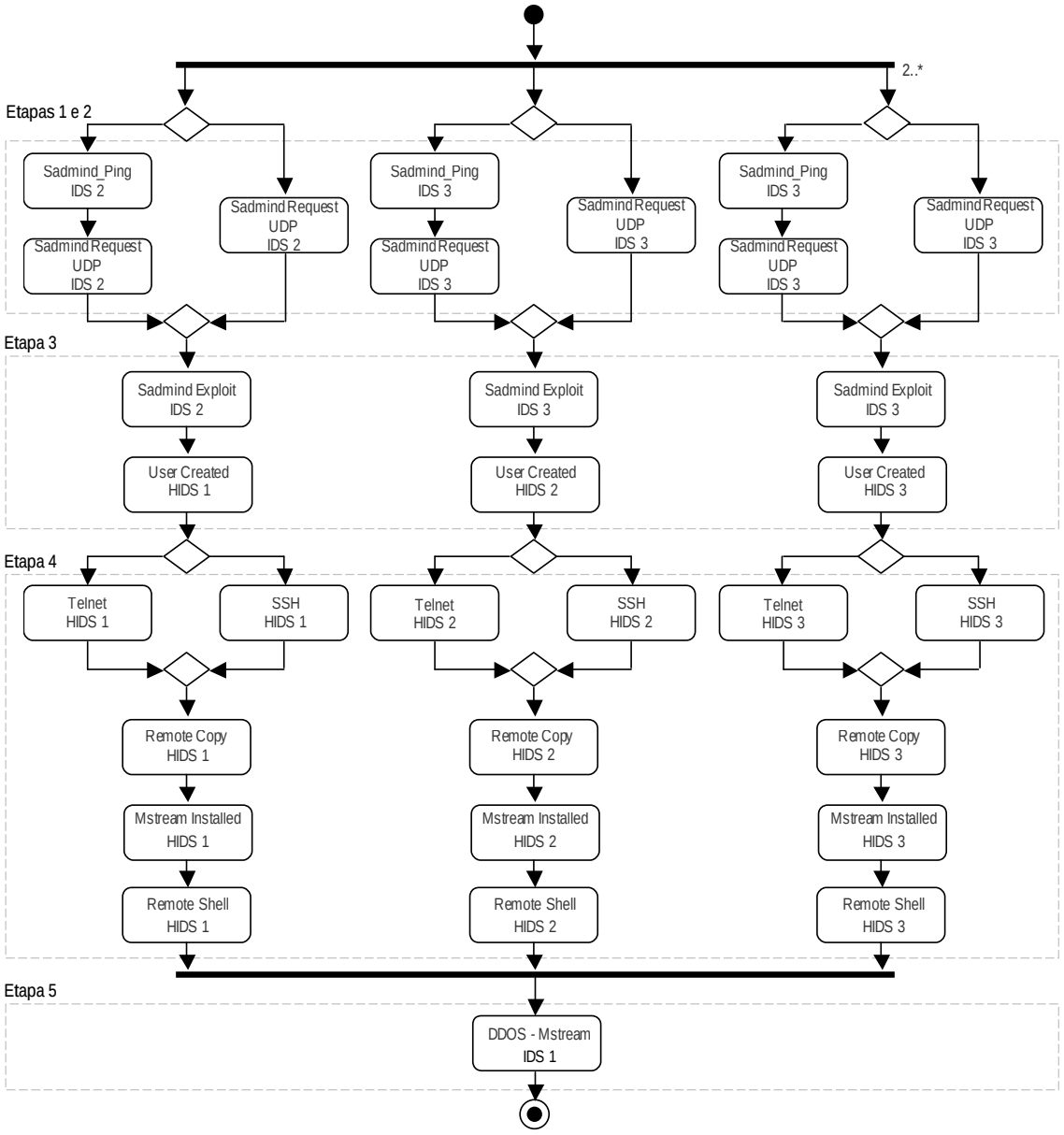

Figura 8 - Representação gráfica de ataque DDoS

- *Instalação do software Mstream DDoS*: esta etapa inicia no instante em que o atacante se conecta (via *telnet* ou *ssh*) aos *hosts* nos quais foram criados novos usuários. Em seguida, utiliza o comando *rcp* para copiar os arquivos binários do software *Mstream DDoS* para esses *hosts*. Após realizar a instalação do software

em cada um dos *hosts* comprometidos, o atacante utiliza o comando *rsh* para inicializar os serviços instalados;

- *Início do ataque*: esta é a fase final, na qual o atacante envia um comando para cada um dos *hosts* comprometidos que, por sua vez, passam a gerar um grande número de pacotes destinados a uma determinada estação alvo.

A Figura 8 ilustra a representação em G-MADL do ataque LLDOS. A primeira etapa é detectada pela observação do evento Sadmin_Ping e a segunda, pelo evento Sadmin Request UDP. A terceira etapa, exploração de vulnerabilidade do serviço *sadmind*, é detectada pela observação dos eventos Sadmin Exploit e User Created. Já a quarta etapa é identificada a partir dos eventos Telnet ou SSH, Remote Copy, Mstream Installed e Remote Shell. Por fim, a detecção da última etapa do cenário exige a observação do evento DDOS Mstream. Destaca-se que os serviços de segurança responsáveis pela observação dos eventos são indicados no rótulo das atividades que representam cada evento.

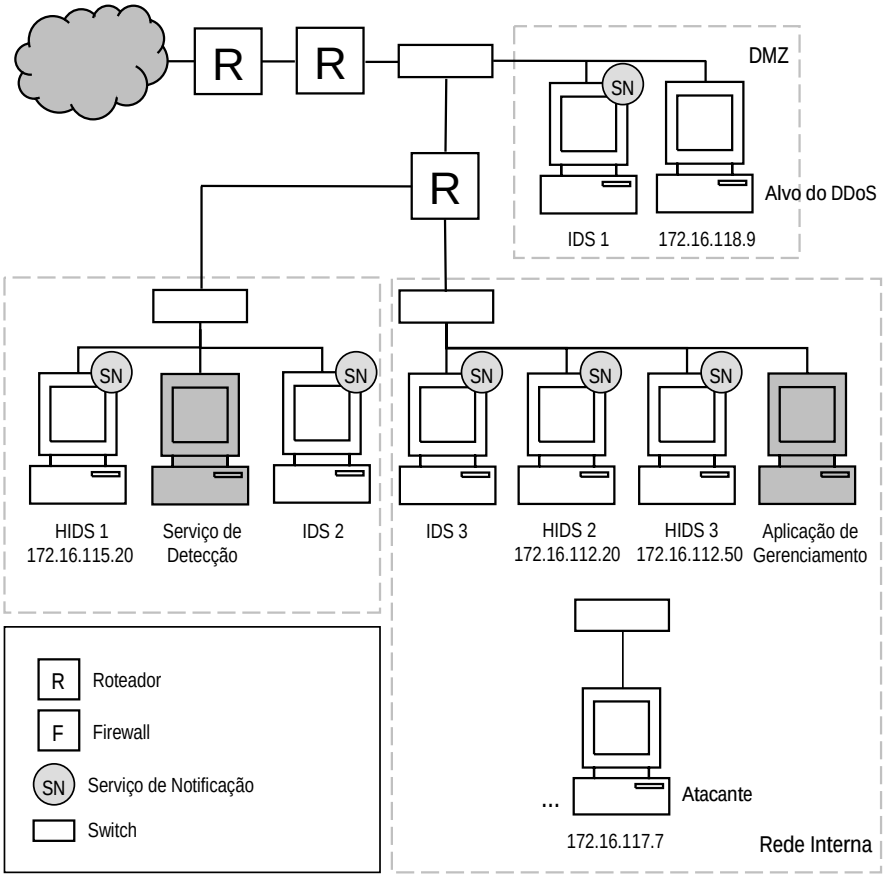

Figura 9 - Infra-estrutura empregada na avaliação experimental

Após a modelagem do ataque, instanciou-se uma estrutura de rede semelhante àquela empregada pelo *MIT Lincoln Laboratory*. Sobre essa estrutura foram instalados aplicação de gerenciamento, serviços de notificação e serviço de detecção, dispostos como ilustra a Figura 9. O passo seguinte consistiu na configuração da arquitetura com o cenário de ataque; as subscrições realizadas junto a cada serviço de notificação estão sintetizadas na Tabela 1. Por fim, reproduziu-se o ataque e acompanhou-se o processo de detecção do mesmo pela arquitetura.

O ataque LLDOS foi detectado por SecCompose. Notificações que não representavam atividades previstas no cenário monitorado, artificialmente enviadas ao serviço de detecção, foram corretamente descartadas pelo mesmo. De forma análoga, notificações recebidas pelo serviço de detecção em ordem diferente daquela definida no cenário de ataque também foram ignoradas.

Tabela 1 - Serviços de notificação e subscrições realizadas

Etapa	Eventos observados	Serviços de notificação
1	Sadmind_Ping	IDS 1, IDS 2 e IDS 3
2	Sadmind Request UDP	IDS 1, IDS 2 e IDS 3
3	Sadmind Exploit	IDS 1, IDS 2 e IDS 3
	User Created	HIDS 1, HIDS 2 e HIDS 3
4	Telnet	HIDS 1, HIDS 2 e HIDS 3
	SSH	HIDS 1, HIDS 2 e HIDS 3
	Remote Copy	HIDS 1, HIDS 2 e HIDS 3
	Mstream Installed	HIDS 1, HIDS 2 e HIDS 3
	Remote Shell	HIDS 1, HIDS 2 e HIDS 3
5	DDOS Mstream	IDS 1

5. Conclusões e Trabalhos Futuros

Este artigo apresentou SecCompose, uma arquitetura de software orientada a serviços para detectar ataques distribuídos e de múltiplas etapas, acompanhada de uma linguagem para modelar essa natureza de ataques. Esta linguagem, denominada *Multistage Attack Description Language* (MADL), possui uma notação gráfica para representação visual e em alto nível do ataque e uma notação textual, baseada em XML, que permite a especificação detalhada do mesmo. A arquitetura, por sua vez, oferece um mecanismo uniforme para comunicação com diferentes serviços de segurança, o que possibilita assinar pelos eventos que constituem os cenários especificados, detectar a sua ocorrência e acompanhar a evolução desses cenários. A arquitetura prevê, ainda, a execução de serviços de contenção que, uma vez invocados, realizam procedimentos para impedir a realização das demais etapas de um determinado cenário de ataque.

SecCompose foi implementada e avaliada, ainda que de forma não exaustiva, usando como base um ambiente próximo do real. A experiência realizada revela que algumas das principais limitações presentes em trabalhos relacionados foram superadas. Primeiro, o emprego de serviços *web* possibilitou a comunicação uniforme com diferentes mecanismos de segurança, em particular – neste trabalho – para obter eventos relevantes. Apesar de oferecer suporte à invocação de ações de contenção, esta possibilidade não foi explorada neste trabalho. Segundo, a detecção de cenários de ataque é realizada *on-the-fly*, característica que será melhor aproveitada quando serviços de contenção estiverem disponíveis e puderem ser invocados para interromper ataques em curso. Terceiro, a linguagem proposta, além de intuitiva (por ser uma derivação dos diagramas de atividades da UML), mostrou-se adequada para a modelagem de um bom número de cenários complexos de ataque.

Como trabalhos futuros mais imediatos pretende-se: (*i*) desenvolver serviços de contenção para alguns mecanismos de segurança como *firewalls*; (*ii*) desenvolver um sistema para geração automática de serviços de notificação e (*iii*) desenvolver uma *Graphical User Interface* (GUI) para a aplicação de gerenciamento com o objetivo de simplificar a especificação e a instanciação de cenários de ataque. Numa perspectiva de

médio prazo, a intenção é generalizar a arquitetura para que toda a questão da segurança nas organizações possa ser expressa na forma de composição e orquestração de serviços.

Referências

Cuppens, F. and Ortalo, R. (2000) “LAMBDA: A Language to Model a Database for Detection of Attacks”. Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID), 197–216.

Cuppens, F. and Miége, A. (2002) “Alert Correlation in a Cooperative Intrusion Detection Framework”. Proceedings of the IEEE Symposium on Security and Privacy, p. 187–200.

Cheung, S., Lindqvist, U., and Fong, W. M. (2003) “Modeling Multistep Cyber Attacks for Scenario Recognition”. Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX III), p. 284–292.

DARPA. (2000) “DARPA Intrusion Detection Scenario Specific DataSets”. http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html.

Debar, H. and Wespi, A. (2001) “Aggregation and Correlation of Intrusion-Detection Alerts”. Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID), p. 85–103.

Dittrich, D. (1999) “The Tribe Flood Network Distributed Denial of Service Attack Tool”. http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt.

Dittrich, D., Weaver, G. and Long N. (2000) “The 'mstream' Distributed Denial of Service Attack Tool”. http://staff.washington.edu/dittrich/misc.mstream.analysis.txt.

Eckmann, T. S., Vigna, G., and Kemmerer, A. R. (2002) “STATL: An Attack Language for State-based Intrusion Detection”. Journal of Computer Security, v. 10, n. 2, p. 71–104.

Fagundes, L. (2006) “Uma Abordagem para Detecção de Ataques Distribuídos e de Múltiplas Etapas baseada na Composição de Serviços Web voltados à Segurança”. Dissertação (Mestrado em Computação Aplicada), Universidade do Vale do Rio dos Sinos.

Fagundes, L. and Gaspary, L. (2005) “Uma Abordagem para Detecção de Ataques Distribuídos e de Múltiplas Etapas baseada na Composição de Serviços Web voltados à Segurança”. Anais do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg), p. 343–346.

Gaspary, L., Meneghetti, E., Sanchez, R., and Antunes, D. (2005) “A SNMP-Based Platform for Distributed Stateful Intrusion Detection in Enterprise Networks”. IEEE Journal on Selected Areas in Communications, v. 23, n. 10, p. 1973–1982.

Graham, S. et al. (2004) Web Services Base Notification. ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-BaseN.pdf.

Ning, P., Cui, Y. and Reeves, D. (2002) “Analyzing Intensive Intrusion Alerts via Correlation”. Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID), p. 74–94.

Northcutt, S. (2000) “Como Detectar Invasão em Rede – Um Guia para Analistas”. Rio de Janeiro: Editora Ciência Moderna.

Pubscribe. (2006) Apache - Web Services - Pubscribe. http://ws.apache.org/pubscribe/.

Porras, A. P., Fong, W. M., and Valdes, A. (2002) “A Mission-Impact-Based Approach to INFOSEC Alarm Correlation”. Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID), p. 95 – 114.

Prelude. (2006) Prelude Homepage. http://www.prelude-ids.org/.

Snort. (2006) Snort Homepage. http://www.snort.org/.

Vambenepe et al. (2005) Management Using Web Services (WSDM-MUWS). Version 1.0. OASIS Standard. http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part1-1.0.pdf.