Especificação formal e Implementação de mecanismos de segurança para a resolução de nomes no DNS

Vagner Sacramento, Anamaria Moreira, Guido Lemos e Thais Batista
Departamento de Informática e Matemática Aplicada - DIMAp
Universidade Federal do Rio Grande do Norte - UFRN
Campus Universitário - Lagoa Nova - 59072-970, Natal - RN
[vagner,anamaria,guido,thais]@dimap.ufrn.br

Resumo

A especificação de um dos protocolos mais importante da internet - DNS (*Domain Name System*) - apresenta falhas que possibilitam a realização de ataques nas aplicações implementadas com base nesta especificação. Um ataque que pode ser aplicado ao DNS é a falsificação de IP na resolução de nomes de domínio, o DNS SPOOFING. Na especificação do DNS não existe nenhuma pré-condição definida para prover segurança contra ataques de DNS SPOOFING. Neste trabalho utilizamos a linguagem B para especificar formalmente parte da RFC 1035 (DNS). Com base nesta especificação mostramos que o DNS apresenta falhas, e especificamos uma solução que propõe medidas de segurança para os problemas encontrados na resolução de nomes no DNS. Como estudo de caso, implementamos os mecanismos de segurança do modelo projetado no bind 8.2.5 e realizamos vários experimentos, implementando ataques de DNS SPOOFING contra o bind 8.2.5 com os mecanismos de segurança e contra o bind 8.2.5 e 9.1.3 sem os mecanismos de segurança. Os resultados obtidos atestam o aumento da resistência a ataques DNS Spoofing da extensão proposta.

`Palavras-chave:` Especificação formal, Vulnerabilidades do DNS, DNS SPOOFING, Mecanismos de segurança para o DNS

Abstract

The specification of one of the most important protocols of the Internet - DNS (Domain Name System) - it presents flaws that allow the accomplishment of attacks in the applications implemented based on this specification. An attack that can be applied to the DNS is the falsification of IP in the domain names resolution, more known as DNS SPOOFING. In DNS specification does not exist precondition defined to provide security against DNS SPOOFING attacks. In this work we used the B language to specify part of RFC 1035 formally (DNS). With base in this specification we proved that DNS possess flaws, and specified one solution that proposes security procedure for the problems found in the DNS names resolution. As case study, we implemented the security patch of the model projected in the bind 8.2.5 and performed several experiments, implementing DNS SPOOFING attacks against the bind 8.2.5 with the security patch and against the original bind 8.2.5 and 9.1.3.

`Keywords:` Formal Specification, DNS vulnerability, DNS SPOOFING, security mechanisms to the DNS

1 Introdução

O Sistema de Nomes de Domínio (DNS) é responsável por traduzir nomes de domínios existentes para endereços IP, atendendo requisições de clientes DNS (resolvers) [2]. O DNS gerencia uma base de dados distribuída, dando suporte a uma variedade de aplicações tais como, servidores de correio eletrônico, servidores Web (www), login remoto (rlogin ou telnet), ftp e outros.

Analisando a especificação da RFC 1035 [2] que define o protocolo DNS, identificamos falhas que podem comprometer a segurança das aplicações que são implementadas com base na RFC 1035. Tal especificação não apresenta soluções eficazes para prevenção de ataques de DNS SPOOFING.

Usando a linguagem B [6], modelamos parte da especificação RFC 1035 para comprovar essa falha. Em seguida, especificamos e implementamos uma extensão do DNS que define medidas de segurança para os problemas encontrados na resolução de nomes. As especificações realizadas foram verificadas com a ferramenta AtelierB [5].

O DNS SPOOFING é uma técnica de ataque que explora as facilidades de predição dos IDs utilizados nas diferentes requisições dos servidores DNS. Historicamente esta técnica de ataque foi introduzida por [19], que relatou as possibilidades de predição de ID na resolução de nomes dos servidores DNS e as inúmeras conseqüências que podem ser exploradas a partir da concretização deste ataque. Utilizamos a aplicação bind no estudo de caso, por ser a mais utilizada na Internet para implementar o protocolo DNS. Vários problemas de segurança relacionados a esta aplicação são discutidos em [20], [21] e [22].

Ataques deste gênero podem ser aplicados a partir de máquinas que estejam no mesmo segmento de rede que o servidor DNS vítima, ou pode ser aplicado a partir de máquinas de redes diferentes da rede que se encontra o servidor DNS vítima. Denominamos o primeiro caso de ataque local e, o segundo de ataque remoto.

Como estudo de caso, implementamos o modelo projetado dentro do código da aplicação bind 8.2.5 [17]. Para testar a eficiência das soluções de segurança projetadas, implementamos uma aplicação que aplica ataques de DNS SPOOFING localmente e remotamente. Realizamos vários experimentos implementando ataques contra o bind com e sem os mecanismos de segurança. Na subseção 5.2 apresentamos os resultados de tais experimentos. Utilizamos o bind da categoria 8.2.* como estudo de caso por ser a versão corrente e recomendada pelo ISC na fase inicial deste trabalho, o bind da categoria 9.* era um projeto em fase de estudo e testes, atualmente o bind desta categoria é a versao recomendada pelo ISC, porém, o bind 8.2.* é o mais utilizado na Internet.

O bind com o *patch* de correção contra ataques de DNS SPOOFING pode ser encontrado no endereço http://www.natalnet.br/~vagner/dns/anti-spoofing.html .

O texto que segue está assim estruturado. A seção 2 apresenta brevemente a estrutura e o funcionamento do serviço de nomes, a aplicação da falsificação de IP em servidores DNS e exemplifica algumas conseqüências desta falsificação. A seção 3 descreve a especificação formal [4] do DNS, acompanhada de uma rápida descrição da estrutura de especificações B. A seção 4 apresenta extensões de segurança para a resolução de nomes no DNS. Nesta seção apresentamos primeiramente uma especificação abstrata seguida de duas outras: uma primeira que pode ser considerada um refinamento da máquina abstrata inicial, e uma segunda especificação que modela algo mais próximo da implementação. A seção 5 contém os detalhes da implementação e os resultados dos mecanismos de defesa inseridos no bind 8.2.5. A seção 6 apresenta os trabalhos correlatos. A seção 7 contém as conclusões desse trabalho.

2 Sistema de Nomes de Domínios - DNS

2.1 Funcionamento do protocolo DNS

O DNS gerencia uma base de dados hierárquica e distribuída indexada por nomes. Um nome de domínio ou uma zona (por exemplo, natalnet.br) possui uma estrutura que reflete a sua colocação dentro da hierarquia de nomes [2].

Para que o processo de resolução de nomes seja uma tarefa rápida, os servidores DNS armazenam o resultado das requisições resolvidas em *cache*. Cada resposta recebida por um servidor é armazenada em *cache* por um determinado tempo, dependendo do tempo de vida atribuído a mensagem (*time-to-live-TTL*). Quando termina a validade, o nome é removido do *cache*. Caso o mesmo nome de domínio seja questionado duas vezes a um servidor DNS, este poderá responder imediatamente a segunda requisição, uma vez que já possui a informação armazenada.

2.2 "SPOOFING" de IP em Servidores DNS

A falsificação de IP na resolução de nomes de domínios -DNS SPOOFING - ocorre quando é enviada uma resposta falsa a uma requisição de resolução de nome para um servidor DNS. Caso seja bem sucedido, o ataque faz com que o servidor armazene em *cache* um endereço IP falso para um determinado nome de domínio[3].

Quando um servidor DNS faz uma pergunta a outro servidor, questionando o endereço IP de um determinado nome de domínio, ele fica aguardando uma resposta. Caso receba uma resposta, ele verifica se determinadas informações estão corretas e, em caso positivo, armazena a resposta em seu *cache*. Uma das informações verificadas na validação das respostas, é o campo ID (identificador) do cabeçalho do pacote DNS. Mais especificamente, a única exigência de autenticação definida na (RFC 1035) é que a resposta DNS deve conter o mesmo valor do ID enviado na requisição. Outros detalhes de segurança necessários para autenticar uma resposta DNS ficam a critério de cada aplicação que implementa este protocolo. Existem aplicações como o DNS SERVER da Microsoft [16] que só utilizam o ID para verificar se uma resposta é válida, conforme exigido pela especificação. Existem outras aplicações como o bind (*Berkeley Internet Name Domain*) [17] que, no intuito de aumentar a segurança validam outras informações (IP, Portas UDP) da resposta recebida, além do ID.

As aplicações que implementam as funcionalidades do protocolo DNS, executando apenas as verificações exigidas na especificação RFC 1035, são vulneráveis a ataques de DNS SPOOFING uma vez que o único teste de autenticidade das respostas baseia-se na verificação do identificador numérico (ID) de 16 bits. Ou seja, para cada resposta recebida deverá existir uma requisição com o mesmo ID. Desta forma, percebemos que cada requisição de resolução de nome enviada estará associada a um identificador que pode variar de 1 até 65535. Teoricamente, um ataque de força bruta pode ser aplicado enviando 65535 pacotes de respostas de resolução de nome com diferentes IDs. Na pior das hipóteses a probabilidade de sucesso seria: 1/65535 x número de tentativas.

2.3 Conseqüências do ataque DNS SPOOFING

Quando o ataque DNS SPOOFING ocorre com êxito, o *cache* do servidor DNS permanece poluído por um tempo, armazenando uma informação incorreta sobre o nome de domínio utilizado no ataque. Várias conseqüências poderão ser exploradas a partir da aplicação deste ataque. Algumas delas são:

1ª) Personificar o conteúdo de uma informação requisitada na abertura de uma página web.

2ª) Driblar mecanismos de defesa que utilizam o nome dos hosts para verificar a autenticidade entre eles.

3ª) Atacar aplicações (e-mail, web, rlogin, telnet, ssh, ftp,...) que necessitam do serviço DNS para tradução de nomes.

3 Especificação do DNS (RFC 1035)

A especificação RFC1035 do DNS se baseia implicitamente na hipótese de que a primeira resposta recebida a uma consulta é verdadeira. Caso seja recebida antes da resposta correta uma resposta falsa que contenha a informação de autenticidade esperada (o identificador da requisição), o servidor DNS irá armazená-la em *cache*. As respostas seguintes serão ignoradas. Conseqüentemente, esta informação falsa será propagada para outros clientes DNS que requisitarem a resolução do mesmo nome de domínio.

Para mostrar formalmente a vulnerabilidade da especificação do DNS descrita na RFC 1035, fizemos uma especificação formal da parte correspondente ao processamento de respostas realizado pelo servidor. Usamos a notação B [6] para descrever os diferentes estados em que o DNS pode se encontrar no processamento das respostas recebidas. Para concentrar nossa análise na possibilidade de ataque por SPOOFING, abstraímos o formato das mensagens e as tabelas para visualizar mais facilmente os dados relevantes.

A notação B é fundamentada na teoria de conjuntos, utilizando boa parte das notações da sua predecessora Z [7], e na *Abstract Machine Notation (AMN)*, que introduz a estruturação da especificação em máquinas de estado, as máquinas abstratas, e permite o uso de notações com estilo imperativo, como atribuições. Alguns elementos necessários para especificar uma máquina (módulo de especificação) em B são descritos a seguir, a partir das próprias especificações do DNS. Os elementos comuns à linguagem Z, são em geral bastante simples (operações sobre conjuntos, relações e funções) e assumidos como conhecidos pelo leitor.

Apresentamos primeiramente a especificação B correspondente à especificação do DNS descrita na RFC 1035. O início da máquina é identificado com a cláusula `MACHINE`, na qual deve ser especificado o nome da máquina (no caso, DNS) e o nome dos parâmetros recebidos, a partir dos quais será construída a especificação. Na máquina DNS temos: `NAME`, conjunto de nomes válidos; `IP`, conjunto de identificadores de IP válidos; e `CONTENT` outras informações sobre o conteúdo de uma resposta ou de uma pergunta, que não são relevantes para a observação do SPOOFING.

MACHINE
```
    DNS(NAME,IP,CONTENT)
```

Para identificar a presença de uma informação falsa, modelamos como uma grande tabela (`tbname_true`) todas as associações verdadeiras de nomes aos respectivos endereços IP de todos domínios existentes na Internet. Sendo assim, em uma situação normal, os *caches* de todos servidores DNS da Internet deverão espelhar partes dessa tabela, ou seja, usando a terminologia da teoria dos conjuntos (base da linguagem B), estar contidos nela. Se por algum motivo o *cache* de algum servidor contiver uma informação que não pertença a esta tabela, identifica-se que este está armazenando uma resposta falsa. A tabela `tbname_true` é apenas um modelo representando as verdadeiras associações entre o par NOME-IP da Internet, e nao será implementada. A cláusula `INCLUDES` permite que a máquina DNS inclua essa especificação, obtendo assim acesso a `tbname_true`.

INCLUDES
    `TBNAME_TRUE(NAME,IP,CONTENT)`

Nas cláusulas `VARIABLES` e `INVARIANT` são declaradas as variáveis que são componentes do modelo, com seus tipos e quaisquer restrições adicionais. A declaração dos tipos das variáveis no invariante sempre é definida através de um predicado, caracterizando os estados válidos do sistema. Declaramos abaixo duas variáveis para representar os estados do DNS: `tbquery` e `tbname`. `tbquery` armazena todas as requisições de resolução de nomes enviadas pelo DNS e ainda pendentes, e `tbname` simboliza o *cache* do DNS, armazenando as resoluções de nomes efetuadas. No invariante definimos que cada identificador está associado ao conteúdo de uma requisição (em `tbquery`), o *cache* do DNS (`tbname`) mapeia nomes às demais informações associadas a eles, e este *cache* sempre deve ser verdadeiro, pois é um subconjunto do conjunto que contém todas possíveis associações (`NOME/IP`) verdadeiras (`tbname_true`). As cláusulas `CONSTANTS` e `PROPERTIES`, declaram e definem identificadores para melhor mantenabilidade da especificação, e a inicialização é feita com as tabelas do DNS vazias.

CONSTANTS
    `maxid, IDQ`
PROPERTIES
    `maxid = 65535` $\wedge$ `IDQ = 1..maxid`
VARIABLES
    `tbquery, tbname`
INVARIANT
    `tbquery` $\in$ `IDQ` $\twoheadrightarrow$ `CONTENT` $\wedge$
    `tbname` $\in$ `NAME` $\twoheadrightarrow$ `(IP * CONTENT)` $\wedge$ `tbname` $\subseteq$ `tbname_true`
INITIALISATION
    `tbquery, tbname :=` $\emptyset,\emptyset$

Na cláusula `OPERATIONS` devem ser especificadas as operações que irão manipular as variáveis que definem o estado da máquina. Todas as operações definidas devem preservar a validade do predicado definido no invariante, ou seja, levar o sistema de estado(s) válido(s) a estado(s) válido(s), caso suas pré-condições de aplicação (cláusula `PRE`) sejam respeitadas. É de responsabilidade do usuário da operação garantir que as suas pré-condições são válidas antes de executá-la. A nossa primeira operação, a operação para gerar a requisição `(generate_query)` é composta por três ações principais: gerar um identificador pertencente ao conjunto de IDs válidos e que não foi atribuído a nenhuma outra requisição pendente, enviar a nova requisição e inclui-la no conjunto de requisições pendentes.

OPERATIONS
sendquery $\leftarrow$ **generate_query(content)** $=$
PRE
    `content` $\in$ `CONTENT`
THEN

ANY idq WHERE
    $\texttt{idq} \in \texttt{IDQ} \wedge \texttt{idq} \notin \texttt{dom(tbquery)}$
THEN
    `sendquery := (idq,content)` || tbquery(idq) := content
END
END;

A operação para receber resposta (`receive_reply`) verifica se existe uma requisição correspondente à resposta recebida (mesmo identificador), e, caso afirmativo, remove o registro da tabela de requisições pendentes, registrando a resposta recebida no *cache*. Caso contrário (identificador inválido), a resposta é ignorada.

receive_reply(idq,name,ip,content) =
PRE
    $\texttt{idq} \in \texttt{IDQ} \wedge \texttt{name} \in \texttt{NAME} \wedge \texttt{ip} \in \texttt{IP} \wedge \texttt{content} \in \texttt{CONTENT} \wedge$
    $\texttt{idq} \in \texttt{dom(tbquery)} \Rightarrow \texttt{name} \mapsto \texttt{(ip,content)} \in \texttt{tbname\_true}$
THEN
  IF $\texttt{idq} \in \texttt{dom(tbquery)} \wedge \texttt{name} \notin \texttt{dom(tbname)}$
  THEN
      $\texttt{tbquery := \{idq\}} \lhd \texttt{tbquery} \parallel \texttt{tbname(name) := (ip,content)}$
  END
END

É importante observar que a pré-condição de `receive_reply` contém, além das informações de tipagem dos parâmetros da operação $\texttt{idq} \in \texttt{IDQ} \wedge \texttt{name} \in \texttt{NAME} \wedge \texttt{ip} \in \texttt{IP} \wedge \texttt{content} \in \texttt{CONTENT}$, uma condição que exige que a resposta recebida seja verdadeira $\texttt{name} \mapsto \texttt{(ip,content)} \in \texttt{tbname\_true}$. Essa condição é essencial para o bom funcionamento do servidor, como explicado na apresentação informal do DNS, mas fica apenas implícita na especificação RFC 1035. Na especificação formal, somos obrigados a explicitá-la, ou não poderíamos provar que a execução de `receive_reply` preserva o invariante do sistema. Essa é uma das vantagens em termos de clareza da especificação formal em relação à informal.

4 Estendendo o DNS para prevenção de ataques DNS SPOOFING

Em um ataque de DNS SPOOFING o servidor DNS vítima normalmente receberá tanto a resposta falsa enviada pelo atacante quanto a resposta verdadeira enviada pelo servidor DNS verdadeiro. Para modelarmos uma solução de defesa, baseamo-nos na afirmação de que em nenhuma outra circunstância, a não ser em uma tentativa de ataque, um servidor DNS receberá simultaneamente duas respostas com o mesmo identificador para o mesmo nome de domínio, apresentando resoluções de nome diferentes.

Para descrever as especificações que modelam a resolução de nomes no DNS de forma mais segura, especificamos três máquinas diferentes. A primeira especificação, mais abstrata, trata as respostas recebidas como um conjunto de respostas, abstraindo a ordem de chegada e questões temporais no recebimento das mesmas. A segunda especificação descreve a utilização de uma tabela auxiliar que armazena as respostas recebidas por um determinado intervalo de tempo, na tentativa de identificar um ataque de DNS SPOOFING. Após este período, as respostas são transferidas para o *cache* do DNS. Já a terceira especificação modelada, armazena a resposta recebida paralelamente no *cache* e na tabela auxiliar. Se durante um determinado intervalo de tempo não chegar nenhuma outra resposta que identifique uma tentativa de ataque, a resposta em questão é removida da tabela auxiliar.

As especificações apresentadas abaixo seguem parte da especificação original, porém, incorporando as extensões para detecção de ataques de DNS SPOOFING.

4.1 Especificação de uma extensão do DNS contra ataques DNS SPOOFING

Esta especificação modela uma máquina onde a operação de tratamento de respostas recebe como parâmetro um conjunto de respostas para a resolução de um mesmo nome de domínio. Cada resposta recebida possui um endereço IP equivalente ao nome de domínio resolvido. Respostas idênticas são tratadas como uma única resposta, dada a modelagem em um conjunto.

O invariante e as variáveis definidas nesta especificação são iguais aos da especificação original do DNS (**seção 3**). A única alteração diz respeito à operação de recebimento/ tratamento das respostas, mostrada abaixo. Ressaltamos que nessa nova especificação precisamos de mais informações sobre o conteúdo de cada resposta para podermos distinguir respostas duplicadas (funcionamento normal) de respostas diferentes (ataque). Estas são definidas no conjunto `REPLIES`.

```
DEFINITIONS
    REPLIES = IDQ ⇸ (NAME*(IP*CONTENT))
```

A operação `receive_reply` é então modificada de maneira a, caso exista uma requisição correspondente às respostas recebidas (mesmo ID), verificar se elas são todas iguais, e apenas nesse caso é removido o registro em questao da tabela de requisições pendentes, sendo posteriormente adicionado a resposta recebida no *cache*. Caso contrário, um ataque é identificado. Como na especificação precedente, se o identificador não corresponde a uma requisição pendente, a resposta recebida é ignorada.

```
resp <-- receive_reply(idq,conj) =
PRE
    conj ∈ REPLIES ∧  dom conj = {idq} ∧
    idq ∈ dom(tbquery) ∧ card(conj) = 1 ⇒ conj(idq) ∈ tbname_true
THEN
    IF (idq ∈ dom(tbquery))  THEN
       IF card(conj) = 1  THEN
           IF dom(ran(conj)) ⊈ dom tbname  THEN
               resp := ip_found ||  tbname := tbname ∪ ran conj ||
               tbquery := {idq} ⩤ tbquery
           END
       ELSE  resp := attack_detected  END
    END
END
```

De acordo com essa modelagem, é apenas necessário que a resposta verdadeira seja recebida para que o ataque possa ser detectado. O invariante que garante a veracidade do *cache* do DNS poderá no entanto ser quebrado na prática caso sejam consideradas limites de tempo no recebimento das repostas. Se supusermos um intervalo de tempo infinito para a coleta das respostas para uma dada consulta, podemos garantir que, em caso de ataque, receberemos tanto a resposta falsa enviada pelo atacante quanto a resposta verdadeira enviada pelo servidor DNS verdadeiro. Porém, se considerarmos apenas um intervalo de tempo pré-definido (finito), é possível que a resposta verdadeira chegue tarde demais. Daí podemos identificar a importância da definição desse intervalo de tempo, do qual falaremos adiante.

4.2 Refinamento da especificação

Refinando a especificação anterior, apresentamos um modelo mais próximo da implementação. Nesta especificação utilizamos uma tabela temporária (`tbname_candidates`) para armazenar durante um tempo finito as respostas recebidas.

Esta tabela temporária, pode ser vista como uma implementação do conjunto de respostas apresentado na especificação anterior. Se após um dado intervalo de tempo, o DNS tiver recebido apenas uma resposta válida (ID válido), descarta-se a possibilidade de uma tentativa de ataque. A resposta será então movida para `tbname` (*cache* do DNS). O invariante definido nesta especificação pode ser quebrado nas mesmas circunstâncias apresentadas para a especificação anterior, ou seja, no caso de a resposta correta chegar após o período de "quarentena" para aquela requisição. A diferença em relação à especificação original, é que a chegada de uma segunda resposta diferente da primeira dentro desse período de quarentena, ativa o processo de detecção do ataque, sem que o *cache* seja poluído com uma informação falsa.

Para essa nova especificação declaramos a tabela auxiliar `tbname_candidates` com o acréscimo do seguinte predicado no invariante:

```
tbname_candidates ∈ NAME ⇸ (IP * CONTENT)
```

A nova operação para receber respostas ficou definida da seguinte forma:

```
resp ← receive_reply(idq,name,ip,content) =
PRE
    idq ∈ IDQ ∧ name ∈ NAME ∧ ip ∈ IP ∧ content ∈ CONTENT
THEN
    IF idq ∈ dom(tbquery) THEN
        IF name ∉ dom(tbname_candidates) THEN
            resp := ip_found || tbname_candidates(name) := (ip,content)
        ELSIF name ↦ (ip,content) ∈ tbname_candidates THEN
            resp := identical_answer
        ELSE
            resp := atack_detected ||
            tbname_candidates := {name} ⩤ tbname_candidates
        END
    END
END;
```

Toda resposta armazenada em `tbname_candidates` que não corresponda a um ataque detectado, deverá ser movida para `tbname`. Abaixo é mostrada a operação desenvolvida para gerenciar a transferência das informações e remover de `tbquery` a requisição equivalente. Apenas a partir deste momento novas respostas à consulta em questão serão ignoradas.

```
table_update(idq,name) =
PRE
    name ∈ dom(tbname_candidates)  ∧  name ∉ dom(tbname) ∧
     idq ∈ dom(tbquery) ∧ name ↦ tbname_candidates(name) ∈ tbname_true
THEN
    tbname_candidates := {name} ⩤ tbname_candidates ||
    tbname(name) := tbname_candidates(name)} ||  tbquery := {idq} ⩤ tbquery
END;
```

4.3 Definição de extensões de segurança com melhor desempenho

Esta especificação utiliza parte das características da especificação original (armazenando a resposta recebida direto no *cache*, podendo enviá-la para clientes DNS) e parte das características definidas na especificação anterior (utiliza uma tabela auxiliar para verificar se a resposta recebida caracteriza uma tentativa de ataque).

Como na especificação original, se a primeira resposta recebida for falsa o invariante de veracidade do DNS é quebrado. Porém, ao contrário da especificação original, basta que a resposta verdadeira chegue (em um intervalo de tempo determinado) para que o ataque seja detectado. Desta forma, mesmo o invariante não sendo preservado durante o intervalo de tempo entre o recebimento da resposta falsa e da resposta verdadeira, após a detecção, o ataque é invalidado e os alertas necessários podem ser gerados.

A operação receber resposta (`receive_reply`) nessa modelagem verifica que não existe nenhuma resposta para o mesmo nome armazenada nas tabelas `tbname` e `tbname_candidates`. A resposta recebida é nesse caso armazenada em ambas. Caso contrário, se a resposta é idêntica à recebida anteriormente, ela é apenas ignorada. O ataque é detectado quando é identificada uma resposta anterior diferente da sendo tratada. Como em todas as especificações precedentes, todo esse procedimento apenas é efetuado caso exista uma requisição pendente com o mesmo identificador.

```
receive_reply(idq,name,ip,content) =
PRE
    idq ∈ IDQ ∧ name ∈ NAME ∧ ip ∈ IP ∧ content ∈ CONTENT ∧
    idq ∈ dom(tbquery) ⇒ name ↦ (ip,content) ∈ tbname_true
THEN
    IF idq ∈ dom(tbquery)  THEN
       IF name ∉ dom(tbname_candidates)   THEN
           IF name ∉ dom(tbname)  THEN
             resp := ip_found || tbname(name) := (ip,content)}||
             tbname_candidates(name) := (ip,content)
           END
       ELSIF  name ↦ (ip,content) ∈ tbname_candidates  THEN
            resp := not_an_atack
       ELSE
          resp := atack_detected ||
          tbname_candidates := {name} ⩤ tbname_candidates
       END
    END
END;
```

5 Implementação e Resultados dos mecanismos de defesa

Como estudo de caso, nos baseamos no modelo formal descrito na seção 4.3. Implementamos o modelo projetado, e inserimos um trecho de código que descreve as extensões de segurança definidas neste modelo dentro da aplicação bind 8.2.5, logo em seguida, recompilamos esta aplicação com as devidas alterações, e realizamos alguns experimentos para validar a eficiência dos mecanismos de segurança implementados.

Além de implementar as soluções de defesa, desenvolvemos uma aplicação que implementa ataques local e remoto nas aplicações bind 8.2.5 e 9.1.3. A partir dos experimentos realizados validamos e comparamos a probabilidade de sucesso do ataque ao bind 8.2.5 com os mecanismos de defesa e sem os mecanismos de defesa, comprovando assim a eficiência das extensões de segurança implementadas. Na subseção 5.1 descrevemos brevemente parte da implementação do modelo de segurança utilizado como estudo de caso, e na subseção 5.2 apresentamos dados estatísticos dos resultados dos mecanismos de defesa implementados. Maiores detalhes sobre a implementação dos mecanismos de defesa implementados podem ser encontrados em [8].

5.1 Implementação das extensões de segurança

De acordo com os requisitos definidos na modelagem apresentada na seção 4.3, realizamos uma análise mais detalhada de todas respostas recebidas. Para realizarmos as análises necessárias acrescentamos outras funcionalidades ao comportamento do bind, criando algumas tabelas e realizando algumas checagem de consistência para simular o modelo projetado.

Dada a modelagem definida para identificar uma tentativa de ataque, é necessário manter uma referência da requisição equivalente à resposta recebida por um determinado intervalo de tempo. Para tanto, criamos uma tabela (`tb_request`) com este propósito que armazena durante um intervalo de tempo uma identificação de todas as requisições enviadas.

A tabela `tb_resp{}` armazena temporariamente as informações de todas as respostas recebidas. Esta tabela está representando a (tbname_candidates) definida na modelagem. Antes de adicionar uma resposta recebida em `tb_resp{}` é verificado se já existe nesta tabela um registro contendo as mesmas informações de autenticidade (ID, Nome, ...) da resposta recebida. Caso seja encontrado um registro armazenado em `tb_resp{}` contendo as mesmas informações de autenticidade da resposta recebida, é identificada a presença de duas respostas de resolução de nome para o mesmo domínio, contendo as mesmas informações de autenticidade. Diante desta circunstância serão analisadas as informações enviadas como resolução do nome de domínio recebidas nas duas respostas identificadas. Se estas informações forem diferentes, é registrada uma tentativa de ataque devido ao fato do servidor DNS ter recebido duas respostas contendo as mesmas informações de autenticidade esperadas com repostas de resolução do nome diferentes.

Na tentativa de detectar e bloquear ataques de DNS SPOOFING, cada resposta recebida é armazenada em `tb_resp{}` por um determinado intervalo de tempo. Só será identificada uma tentativa de ataque se o atacante conseguir enviar uma resposta para o servidor DNS vítima, contendo as informações de autenticidade esperadas por este servidor, ou seja, somente quando o atacante conseguir êxito na aplicação do ataque é que os alertas serão gerados. Desta forma, estamos evitando que circunstâncias não esperadas no funcionamento do DNS gerem alarmes falsos.

Após receber uma resposta, esta será armazenada em `tb_resp{}`, caso não exista nenhuma outra resposta similar (utilizando o mesmo ID) armazenada nesta tabela. Esta resposta permanece sob análise durante um tempo que varia entre 5000 e 9999 ms. Se durante este intervalo de tempo, não chegar nenhuma outra resposta com as mesmas informações de autenticidade esperadas, caracterizando uma tentativa de ataque, o registro em questão será removido de `tb_resp{ }` e de `tb_request{}`. O tempo estimado para análise das respostas DNS é baseado nas estatísticas da performance do DNS mostradas pelo MIT Laboratory for Computer Science e pelo Korea Advance Institute of Science and Technology (KAIST) [18]. As estatísticas apresentadas em [18] mostram que o tempo médio para resolução de um nome é 97 ms, sendo assim, deduz-se que o tempo ($> 5000 <=$ 9999 ms) estipulado para análise de uma resposta para detecção de um ataque é suficiente.

A detecção do ataque tornou-se possível porque a requisição equivalente à resposta recebida não foi removida da tabela de requisições enviadas quando a primeira resposta foi recebida. Tal operação representada no modelo pelo método `table_update(idq,name)` é realizada através de uma função chamada de `clean_table_aux(ID,Nome)`.

Após identificada uma tentativa de ataque, são gerados os alertas necessários:

- Gravar nos logs os detalhes do acontecido;
- Enviar e-mail para o administrador da rede;
- Remover do *cache* do servidor DNS somente o registro que contenha o nome utilizado para aplicar o ataque.

5.2 Resultados dos mecanismos de defesa

Utilizando o bind 8.2.5 com as novas funcionalidades de segurança, obtivemos 100% de sucesso na detecção de ataques locais. No experimento realizado, aplicamos 675 ataques contra domínios diferentes.

Uma comparação da eficiência do ataque local no bind 8.2.5 com as extensões de segurança e outras implementações do protocolo DNS são mostradas na tabela 1.

675 Tentativas de ataques	**BIND 8.2.5 modificado**	**BIND 8.2.5 original**	**BIND 9.1.3 original**
Sucesso	0%	100%	100%

Tabela 1: Resultados da eficiência do ataque local

Os resultados dos mecanismos de defesa para o ataque remoto são mostrados na tabela 2. A tabela 2 mostra os resultados das tentativas de ataque ao bind 8.2.5 modificado, e ao bind 8.2.5 e 9.1.3 original. Nós aplicamos ataques remoto aos servidores DNS do laboratório NatalNet executando bind 8.2.5 e 9.1.3 durante um período de 90 horas.

O bind com os mecanismos de defesa implementados apresentou bons resultados na detecção dos ataques. O bind 9.1.3 dispõe de mecanismos que dificultam a aplicação do ataque, pois este inibe o envio de três ou mais requisições para a resolução do mesmo nome de dominio, corrigindo o problema apresentado por sua antecessora (bind 8.2.*).

BIND	Numero de Tentativas	QTd. De Sucesso	Percentual de Sucesso
8.2.5 Modificado	31752	3	0,013%
8.2.5 Original	31761	1872	5,894%
9.1.3 Original	32752	88	0,270%

Tabela 2: Resultados dos mecanismos de defesa contra ataques de DNS SPOOFING remoto

O ataque ao bind com as novas funcionalidades de defesa não será detectado, caso não sejam recebidas duas respostas para a resolução do mesmo nome de domínio, contendo as mesmas informações válidas esperadas pelo servidor DNS vítima com resoluções de nome diferentes. Como o DNS utiliza o protocolo UDP para enviar as suas requisições, não temos certeza de que a requisição de resolução de nome enviada pelo servidor DNS vítima vai chegar ao seu destino e, mesmo que chegue ao destino, não se tem a certeza de que a resposta enviada pelo servidor DNS questionado retorne ao servidor DNS vítima. O UDP é um protocolo da camada de transporte não confiável que não oferece mecanismos de recuperação dos datagramas perdidos [9].

Na tentativa de amenizar a perda de pacotes UDP, o bind faz um controle de recebimento das respostas. Se nenhuma resposta para uma dada requisição for recebida num intervalo de 4 segundos, ele retransmite uma requisição utilizando o mesmo ID.

Na tentativa de tornar mais difícil a detecção do ataque, o atacante pode tentar aplicar o ataque ao servidor DNS vítima e ao mesmo tempo aplicar um ataque de *flooding*[1] ao suposto servidor DNS que será questionado. O atacante pode tentar imobilizar o DNS verdadeiro fazendo com que ele não envie uma resposta para o servidor DNS vítima impossibilitando a identificação do ataque, pois não será recebida uma segunda resposta. A grande dificuldade do atacante é saber qual é o servidor DNS que ele deverá aplicar o ataque de *flooding*. Como os servidores DNS resolvem nomes de forma interativa, o atacante não sabe para qual servidor o DNS vítima enviou uma requisição. O atacante pode tentar aplicar um *flooding* distribuído direcionado a vários servidores DNS que provavelmente receberam uma requisição de resolução de nome enviada pelo servidor DNS vítima. O grande problema desta estratégia é que dentre estes prováveis servidores que receberam uma requisição estão os ROOT-SERVERS, no entanto, imobilizar estes servidores sem gerar alertas para aplicar um ataque é, na prática, quase impossível. Além de que o atacante terá que aplicar um ataque de *flooding* a cada nova tentativa de ataque.

6 Trabalhos correlatos

6.1 DNS Wrapper

O projeto apresentado em [1] relata a elaboração de um sistema de nomes conhecido como DNS Wrapper.

O DNS Wrapper é uma aplicação modelada e especificada formalmente em VDM que foi incorporada ao código do bind com objetivo de examinar as mensagens de DNS

[1] Flooding (inundação) - este ataque consiste basicamente no envio de um grande número de mensagens, para um determinado servidor, com endereço IP de origem forjado, com intuito de fazer com que a fila de recepção de mensagens em andamento fique lotada.

que estão sendo enviadas e recebidas. Esta aplicação consiste em duas partes principais: Wrapper-q para processar as requisições e Wrapper-r para processar as respostas.

Como estudo de caso, este projeto utilizou o bind 4.9.5 que possui um algoritmo de geração de ID seqüencial. Na tentativa de prevenir ataques de DNS SPOOFING, o DNS Wrapper implementa uma geração aleatória dos IDs a serem enviados nas requisições de resolução de nomes. Desta forma, o DNS Wrapper não leva em conta a possibilidade de ataques locais que capturam o ID corrente utilizado pelo servidor DNS, e não considera a existência de técnicas de ataques específicas, que realizam a predição de IDs aleatórios. Além da estratégia de aleatoriedade dos IDs ser ineficiente, ela já é implementada no bind das categorias 8.2.* e 9.*, não sendo necessário utilizar o DNS Wrapper. No trabalho realizado em [8] foram citadas algumas técnicas que comprovam a aplicação de ataques de DNS SPOOFING em todas versões existentes do bind, independente do tipo de algoritmo de geração de ID que esteja sendo utilizado (seqüencial ou aleatório).

Na modelagem do sistema foi utilizado o conceito de servidores autoritários sobre uma zona, na qual fez-se necessário criar um módulo `AuthServer` que mapeia uma zona para uma lista de servidores autoritários. Este módulo retorna verdadeiro se e somente se todos registros *resource record* de entrada pertencem ao conjunto de registros autoridade. A especificação em VDM que descreve este módulo do sistema é mostrada abaixo.

```
AuthServer :  Zone → Server-set
AuthAnswer :  Idx → RR-set
   ∀i ∈ dom(Db) .  AuthAnswer(i) =
   let z ∈ Zone ∧ p ∈ Process ∧
   i ∈ dom ZoneData ∧ ∧ p ∈ AuthServer(z) in
   View(p)(i)
Authoritative :  RR-set → Boolean
∀rrs ∈ RR-set .  Authoritative(rrs) =
   ∀rr ∈ rrs .  rr ∈
   AuthAnswer((dname(rr), type(rr),class(rr)))
```

O invariante definido em uma máquina, tem que ser formado por um predicado que seja verdadeiro durante a execução do sistema modelado, porém, a especificação definida para o DNS Wrapper não garante a integridade do invariante, porque ela permite que servidores DNS aceitem respostas falsas de servidores que não tem autoridade sob o domínio questionado. Conforme mostrado na especificação do módulo `AuthServer`, o sistema não consegue identificar se um servidor é realmente autoritário sobre uma zona. Além da especificação não preservar o invariante, ela não define nenhuma solução eficaz para detecção e prevenção de ataques de DNS SPOOFING. O invariante definido para esta especificação é mostrado abaixo.

```
state DNS of
   protectedNS:Server-set △
   ...
   inv mk-DNS(protectedNS)
   ∀s ∈ protectedNS . Authoritative(rng View(s))
end
```

Cientes da facilidade de implementação de ataques de DNS SPOOFING, definimos na especificação um invariante que descreve um predicado que possibilita a deteção da presença de uma resposta falsa.

INVARIANT

$\texttt{tbquery} \in \texttt{IDQ} \nrightarrow \texttt{CONTENT} \wedge \texttt{tbname} \in \texttt{NAME} \nrightarrow (\texttt{IP} * \texttt{CONTENT}) \wedge \texttt{tbname} \subseteq \texttt{tbname\_true} \wedge$
$\texttt{tbname\_candidates} \in \texttt{NAME} \nrightarrow (\texttt{IP} * \texttt{CONTENT})$

Ao contrário do DNS Wrapper, propomos uma solução de defesa contra ataques de DNS SPOOFING, que não se baseia no tipo de algoritmo utilizado por uma determinada aplicação que implementa o protocolo DNS. A solução modelada e implementada, apresenta bons resultados na detecção e prevenção de ataques, além de que a estratégia de defesa utilizada pode ser incorporada em qualquer aplicação que implementa o DNS.

6.2 DNSSec

Uma solução que está sendo projetada pelo IETF[2] desde 1994 [10] e ainda não está sendo amplamente usada é o DNSSec [3]. DNSSEC é o nome dado às extensões de segurança que estão sendo propostas para o protocolo DNS definida pela RFC 2535. Atualmente, existem alguns grupos de pesquisa realizando experimentos com as novas características oferecidas pelo DNSSec. Alguns grupos que estão atuando na fase de experimentos, validando as definições transcritas na RFC 2535 são: [12] [13]. Detalhes de como utilizar o DNSSec com o bind 9 podem ser encontrados em [11] e [14].

A RFC 2535 estabelece os detalhes das alterações propostas, para a implantação de um DNS seguro. A característica principal deste novo modelo do DNS é prover autenticação da origem dos dados das zonas e verificar a integridade destes dados usando criptografia de chave pública [3].

Alguns problemas existentes na utilização do DNSSec apresentados por [11] são:

- Performance;
- Deficiente estrutura de atualização de chaves públicas;

Os problemas de performance do DNSSec estão associados ao tempo de uso da CPU e memória para validar assinaturas digitais. Segundo as estatísticas realizadas por [11], o custo de processamento real gerado pela encriptação/decriptação de pacotes DNS depende do algoritmo utilizado. Com DSA-512 é possível assinar aproximadamente 135 dominios/Segundos em um PC de 500 Mhz com FreeBSD e utilizando o RSA-1024 temos uma taxa de aproximadamente 17 domínios/segundos e com o DSA-768 temos em média 62 domínios/segundos.

Outro problema citado em [11] é a necessidade de uma estrutura de distribuição de chaves pública. Um dos motivos para a existência de tal estrutura, é a necessidade de manter a integridade e autenticidade das chaves armazenadas no *cache* dos servidores DNS.

O laboratório NLnet está atualmente executando uma série de experimentos utilizando o DNSSec e os resultados destes experimentos podem ser encontrados em [14].

Como podemos perceber, o DNSSec dispõe de uma série de recursos para prover segurança na comunicação entre servidores DNS. Porém, estes recursos ainda não estão sendo utilizados pela comunidade Internet devido à algumas limitações comentadas anteriormente. O custo de processamento vai ser um dos maiores obstáculos para a implantação do DNSSec. Para termos idéia do custo de processamento de um servidor DNS muito

[2]IETF - Internet Engineering Task Force

solicitado, podemos levar em conta a média de requisições recebidas por um servidor pertencente aos ROOT-SERVERS. Cada servidor recebe em média 272 milhões de requisições por dia, em torno de 1.800.000 por minuto [15]. Se todas estas requisições e suas respectivas respostas fossem criptografadas o tempo de processamento seria consideravelmente aumentado o que torna impraticável o uso do DNS.

Desta forma, percebemos que o DNSSec é uma solução que está na fase de planejamento, testes, experimentos e que no futuro poderá ser utilizado. No entanto, ele não é uma opção que possa ser adotada para remediar os problemas de segurança atualmente encontrados na comunicação entre servidores DNS.

7 Conclusão

Neste artigo apresentamos falhas de segurança na especificação do protocolo DNS transcrita na RFC 1035. Estas falhas permitem que ataques de DNS SPOOFING afetem as aplicações implementadas com base nesta especificação. Para corrigir o problema apresentado na especificação original, modelamos e especificamos formalmente em B, extensões de segurança que podem ser implementadas na resolução de nomes do DNS. A partir das especificações modeladas constatamos que é possível implementar a resolução de nomes no DNS de forma mais segura, fazendo com que as aplicações que implementam este protocolo possam usufruir do modelo de segurança proposto.

A solução implementada no estudo de caso foi incorporada ao código da aplicação bind 8.2.5. A maior dificuldade de incorporar as soluções projetadas ao código do bind foi à necessidade de realizar um estudo do código desta aplicação. O bind 8.2.5 possui aproximadamente 65000 linhas de código totalmente desestruturadas, com muitas instruções complexas que dificultam a interpretação, algumas destas são: goto, inicialização dos campos das estruturas utilizando aritmética de ponteiros, deslocamento de byte, dentre outros. De acordo com os resultados mostrados na tabela 1 e 2, obtivemos excelentes resultados na detecção de ataques de DNS SPOOFING local e remoto. A partir dos experimentos realizados comprovamos que existem facilidades nas aplicações atuais do DNS que permitem que ataques deste gênero sejam aplicados nas versões atuais com alto percentual de sucesso. Os experimentos foram realizados com os servidores DNS do Laboratório NatalNet e com os servidores DNS do laboratório de pós-graduação do DIMAp da UFRN.

Utilizando a linguagem B, especificamos o mecanismo de defesa proposto explorando as características principais desta linguagem, que são: verificação de tipos, realização de provas que validam as consistências dos predicados descritos no invariante, representação matemática do modelo proposto.

Referências

[1] CHEUNG, Steven e LEVITT, Karl N. A Formal-Specification Based Approach for Protecting the Domain Name System. Departamento de Ciência da computação - Universidade da Califórnia. www.cs.ucdavis.edu, 2000-06.

[2] MOCKAPETRIS, P. Domain Names - Implementation and Specifications. STD 13, RFC 1035. November 1987. (URL: ftp://ftp.isi.edu/in-notes/rfc1035.txt)

[3] DAVIDOWICZ, Diane. Domain Name System (DNS) Security. 1999 (URL: http://compsec101.antibozo.net/papers/dnssec/dnssec.html).

[4] POTTER, Ben and SINCLAIR, Jane and TILL, David. An Introduction to Formal Specification and Z. Prentice Hall: Second Edtion, 1996

[5] ABRIAL, Jean-Raymond and GEC Alsthom Transport. ATELIER B.

[6] K LANO: The B language and Method. A guide to Pratical Formal Development. Springer-Verlang, 1996.

[7] SPIVEY, Mike. The Z Notation: A Reference Manual. 2nd edition. Prentice Hall, 1992.

[8] SACRAMENTO, Vagner e BATISTA, Thais e LEMOS, Guido e MONTEIRO, Claudio. Uma Estratégia de defesa contra ataques de DNS SPOOFING. 3 Simpósio de Segurança em Informática. ITA - Instituto Tecnologico de Aeronáutica. São José dos Campos - SP 24 a 26 de Outubro de 2001.

[9] SOARES, Luiz Fernando G. LEMOS, Guido. COLCHER, Sergio. Redes de computadores: das LANs , MANs e WANs às redes ATM. Rio de Janeiro: Campus, 1995.

[10] CHAR. IETF DNSSEC WG. DNS Security (dnssec). Charter. IETF, 1994. (URL:http://www.ietf.cnri.reston.va.us/proceedings/94dec/charters /dnssec-charter.html).

[11] NEMETH, Evi. Securing the DNS. Proc. in USENIX. November 2000. (URL: http://www.usenix.org/publications/login/2000-11/index.html).

[12] DNSEXT. Work Group IETF DNS Extensions Workgroup. 1996. (URL: http://www.ietf.org/ html.charters/dnsext-charter.html).

[13] DNSOP. Work Group IETF Domain Name Server Operations. 1996. (URL: http://www.ietf.org /html.charters/dnsop-charter.html).

[14] NLnet Labs. DNS Resources. Amsterdam, 01 jan, 2000. (URL: http://www.nlnetlabs.nl/dnssec).

[15] ISC F-Root-server. Internet Software Consortiun (ISC). 1998. (URL: http://www.isc.org/services/public/F-root-server.html).

[16] MICROSOFT DNS SERVER - Part of Service Pack 6: http://www.microsoft.com/ntserver/nts/downloa ds/recommended/SP6/allSP6.asp. 1999.

[17] Consortiun (ISC). (URL: http://www.isc.org/bind/).

[18] JUNG, Jaeyeon and SIT, Emil and BALAKRISHNAN, Hari and MORRIS, Robert. DNS Performance and the Effectiveness of Caching. MIT Laboratory for Computer Science. Proc. in ACM SIGCOMM INTERNET MEASUREMENT WORKSHOP. 2001.

[19] VIXIE, Paul. DNS and BIND Security Issues. Proc. Of the 5th UNIX Security Symposium, June 5-7, 1995, pp.209-216.

[20] S. M. BELLOVIN. Using the Domain Name System for System Break-ins. Proc. Of the 5th UNIX Security Symposium, June 5-7, 1995, pp.199-208.

[21] CERT Coordination Center. Topic: Multiple Vulnerabilities in BIND. Advisory CA-98-05, April 8, 1998.

[22] CERT Coordination Center. Topic: Multiple Vulnerabilities in BIND. Advisory CA-2001-01, Janeiro 2001. (URL: http://www.cert.org/advisories/CA-2001-02.html).