Proposta de um novo protocolo anti−spam complementar ao SPF

Danton Nunes, Internexo Ltda. São José dos Campos, SP
danton.nunes@inexo.com.br

Agenda

- Motivação
- Ideias
  - Uso do reverso (in−addr.arpa ou ip6.arpa)
  - Hashes
  - Valores resultantes
  - Exemplos
- Implementação
- Comentários
- Agradecimentos

Motivação

SPF (Sender Policy Framework) serve para o "dono" de um domínio dizer quais são os endereços IP autorizados a enviar mensagens desse domínio, através de uma ACL enviada por DNS.

Atualmente o protocolo é definido pelas RFCs 4408 e 6652.

Resumindo: antes de receber um email de um dado IP e de um dado domínio perguntamos para o "dono" do domínio se o IP é bom.
Com isso podemos decidir aceitar ou não a entrega.

Mas ninguém pediu a opinião do "dono" do endereço!

É exatamente isto que este trabalho propõe, um mecanismo para questionar o "dono" de um bloco de endereços se determinado endereço IP pode ou não enviar email de um dado domínio.

E há bons motivos para isso...

Motivação

STF tem sido abusado por alguns espertalhões:

```
$ host -t txt treinamentosdvw.com.br
treinamentosdvw.com.br descriptive text "v=spf1 a mx include:dc-dvw.com.br +all"
```

<nota>

Casos assim podem ser tratados estendendo o milter de SPF para testar algum endereço que reconhecidamente não pode enviar email, por exemplo, algum endereço local ou multicast.

```
$ spfquery -ip=ff02::2 -sender=envelope-from=qualquer1@treinamentosdvw.com.br
pass
```

Resumindo, o dono desse domínio nos diz que o endereço IPv6 all-routers está autorizado a enviar email desse domínio, o que é obviamente absurdo.

</nota>

Ideias

Uso do reverso (in−addr.arpa ou ip6.arpa)

O responsável por um bloco de endereços IP geralmente controla o domínio de tradução reversa, dentro de in−addr.arpa (IPv4) ou ip6.arpa (IPv6).

Nada mais natural do que usar o domínio de tradução reversa para autorizar ou não o envio de email de um dado domínio.

Assim como no SPF usa−se um RR tipo TXT. No SPF o TXT contém uma lista de controle de acesso. Aqui o TXT carrega a própria resposta.

O nome de domínio (ownername) contém:

- Hash do nome do domínio testado em base32 (RFC−4648)
- Endereço IP em ordem reversa (octetos em decimal p/ IPv4 ou hexa para IPv6)
- in−addr.arpa para IPv6 ou ip6.arpa para IPv6

O texto resultante (rrdata) contém: `v=rspf1 PALAVRA`

`pass`	email do domínio pode sair do endereço IP testado
`fail`	email do domínio NÃO pode sair do endereço IP testado
`neutral`	indiferente (há controvérsias se este valor deve existir)

Ideias

Hashes

To hash or not to hash, that is the question!

PRÓ

O domínio a ser testado deve ser afixado ao domínio w.x.y.z.in−addr.arpa ou (32 hexadecimais).ip6.arpa, e isso pode exceder o número máximo de compo− nentes de um nome de domínio.

Domínios com número fixo de componentes são mais tratáveis por "robozinhos".

CONTRA

Complica o uso de coringas (wildcards) para estabelecer políticas no atacado.

Torna os nomes de domínio resultantes ilegíveis para humanos.

Consenso: usar hash codificado em base 32hex (RFC−4648) sem ’padding’.
Base 16 fica muito comprido e base 64 não é compatível com nome de domínio.

Funções de hash candidatas: MD5, SHA−1, SHA−256
Os primeiros experimentos foram feitos com MD5, mas tudo indica que SHA−256 deverá ser a escolha definitiva (modernidade, implementação em hardware)

Ideias

Valores resultantes

O tipo do RR é TXT. O rrdata resultante contém:

- ’v=rspf1’ indicando o protocolo e sua versão
- espaço(s) em branco como separador
- palavra de resultado que pode ser ’pass’, ’fail’, ou ’neutral’

Maiúsculas e minúsculas tem o mesmo valor em todo o conteúdo.

Qualquer coisa diferente de ’pass’, ’fail’, ou ’neutral’ deve ser entendido como erro.

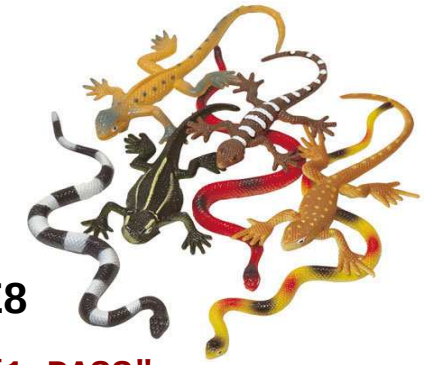
Também é erro haver mais de um TXT começando com ’v=rspf1’ mesmo que tenham o mesmo rrdata.

NXDOMAIN é interpretado como ’neutral’.

Respostas possiveis:	pass	enviar email desse domínio e endereço é OK.
	fail	enviar email desse domínio e proibido.
	neutral	não tenho opinião formada sobre o assunto.
	error	alguma coisa deu errado.

Ideias

Exemplos



Autorizando o IP 192.0.43.10 a enviar email do domínio example.com:

example.com —MD5, base32hex→ **BATBQO1R49S060MTHM1KJ3IHE8**

```
BATBQO1R49S060MTHM1KJ3IHE8.10.43.0.192.in-addr.arpa.  IN  TXT  "v=rspf1 PASS"
```

Proibindo o domínio bad.example.com de mandar email desse mesmo IP:

bad.example.com —MD5, base32hex→ **GJJ7HSS8RME2UJE7KTT7Q8FKSK**

```
GJJ7HSS8RME2UJE7KTT7Q8FKSK.10.43.0.192.in-addr.arpa.  IN  TXT  "v=rspf1 FAIL"
```

Usando coringa (wildcard) para estabelecer uma política padrão para o bloco /24:

```
*.43.0.192.in-addr.arpa.  IN  TXT  "v=rspf1 FAIL"
```

Isto é muito interessante porque permite vetar qualquer domínio e definir as exceções individuais, p.ex. no caso de uma rede DSL com a maioria dos usuários domésticos e alguns corporativos no meio.

Implementação

Acessório para MTAs

Milter usando a libmilter (www.milter.org). Funciona com sendmail e postfix.

Foram implementados os métodos:

- connect, captura o endereço IP do transmissor;
- envfrom, obtém e testa o nome de domínio declarado no MAIL FROM:.

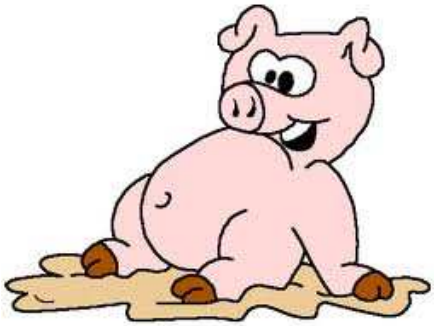
```
chave=base32hex(MD5(remetente.dominio));
ownername=chave+'.'+IPv4?IP.revert_octets()+'.in-addr.arpa':
                       IP.revert_nibbles()+'.ip6.arpa';
rrset=resolv(ownername,'TXT');
if (rrset.status==NXDOMAIN) return NEUTRAL; //política default razoável
if (rrset.status==TIMEOUT) return ERROR;
rspfset=rrset.match(/^v=rspf1 /i);
if (rspfset.length==0) return NEUTRAL;
if (rspfset.length>1) return ERROR;
switch (rspfset[0].toLowerCase()) {
    case 'v=rspf1 pass': return PASS;
    case 'v=rspf1 fail': return FAIL;
    case 'v=rspf1 neutral': return NEUTRAL;
    default: return ERROR;
}
```

Implementação

Estado da coisa

Esta implementação é meio porca, estamos trabalhando em algumas melhorias:

- codificação do método envrcpt que permite uma política por destinatário,
- substituição do MD5 por SHA256,
- refazer o codificador de base32hex, que é 'endian−dependent'.



Além disso falta escrever um programa para facilitar e automatizar o cálculo dos hashes para inscrever os RRs nos arquivos de zona.

No processo de desmporcalhamento vamos separar a função que faz a consulta dos métodos da libmilter, de modo que ela possa ser usada em outras arquiteturas de filtros de email.

Logo que estiver minimamente documentado, o milter será disponibilizado para a lista GTER. (Atualmente ele tem DOIS comentarios!)

O milter foi escrito em C (como nos velhos tempos).

Comentários

Enfim para que serve?

Em uma resposta curta: para o mesmo que o SPF, isto é, validar o uso de um recurso (o endereço IP) por uma aplicação de outra pessoa (o email).

A condição de FAIL pode ser usada tranquilamente para rejeitar a mensagem, ainda na fase de negociação do SMTP.

O resultado ERROR pode ser interpretado como FAIL ou NEUTRAL, de acordo com a vontade do freguês.

Problemas para detentores de blocos IPv4 menores que /24

A BCP−20 poderia ser usada mas é um trambolho e os macaquinhos ainda não estão bem treinados.

http://tools.ietf.org/html/draft−gersch−dnsop−revdns−cidr−02 apresenta uma ideia promissora para resolver os problemas da BCP−20.

Comentários

Problemas para detentores de blocos IPv4 menores que /24 (cont.)

Durante o processo de discussão na lista GTER surgiu uma proposta de uso de chave pública para contornar os problemas de delegação da BCP−20, possivelmente associado ao DNSSEC, mas o consideramos muito complexo para um primeiro rascunho.

Problema de ovo e galinha

É claro que este esquema só funciona se for amplamente aceito. O mesmo vale para o SPF tradicional. Afinal, Tostines está sempre fresquinho porque vende mais ou vende mais porque está sempre fresquinho?

Obviamente, se não começar não vai ser usado mesmo!

Comentários

A fazer

Reescrever o milter, separando a interface da parte funcional, documentando melhor e substituindo MD5 por SHA−256;

Escrever programa ou script para facilitar a escrita dos RRs, com todas as cobras e lagartos em base32hex.

Estudar a questão da delegação para blocos menores que /24. Esse problema não existe em IPv6, pois a zona ip6.arpa é bem mais capilarizada.

Publicar o código desenvolvido para testes, de preferência com uma licença liberal tipo GPL ou BSD.

Testar, testar, testar.

Juntar tudo, traduzir para o inglês e publicar na forma de uma RFC.

Agradecimentos

Aos participantes da lista GTER que contribuiram com esta discussão. São muitos para enumerar aqui. Suas sugestões foram muito importantes para rascunhar o protocolo.

A InterNexo e alguns de seus clientes que involuntariamente serviram de cobaia para os testes.

Perguntas