

RSA[®]CONFERENCE2014

FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Share.
Learn.
Secure.

Capitalizing on
Collective Intelligence

DLL Side-Loading: A Thorn in the Side of the Anti-Virus (AV) Industry

SESSION ID: HTA-W04A

Amanda Stewart

Malware Research Engineer
Fireeye



Overview

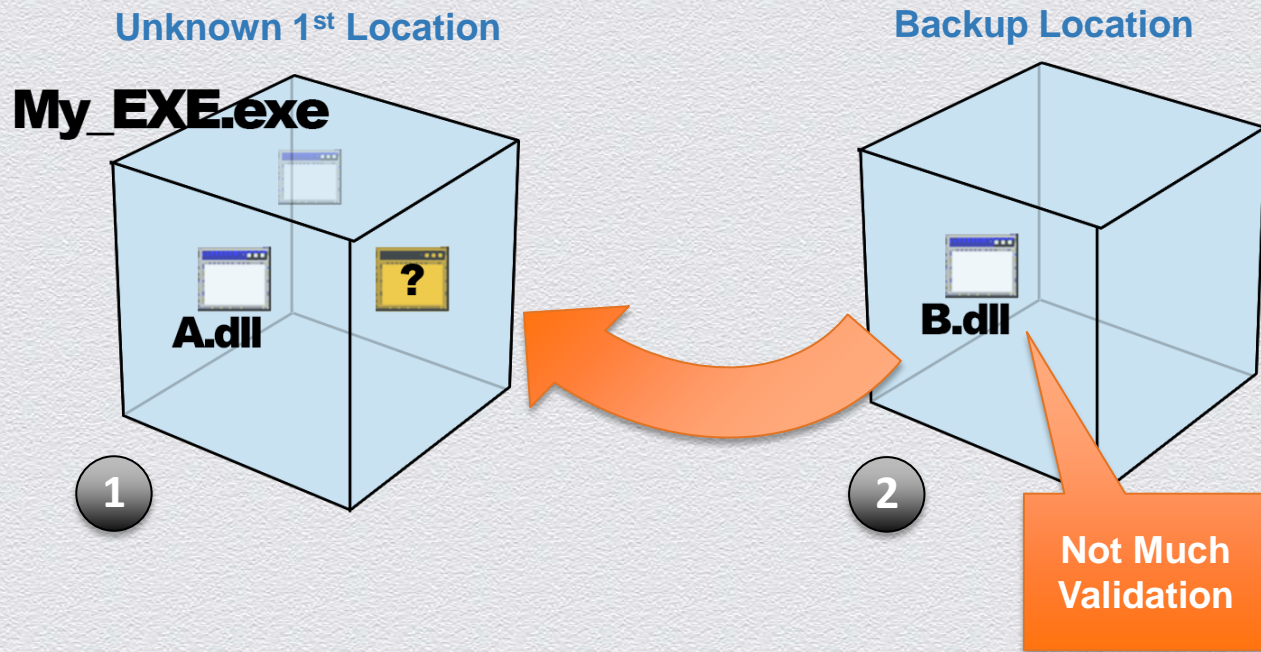
- ◆ What is DLL Side-loading?
- ◆ APT using the technique
- ◆ How to recognize it?
- ◆ How to avoid it?

What is DLL Side-loading?

- ◆ History
 - ◆ Search-Order-Hijacking
 - ◆ (A.K.A: DLL preloading attack or binary planting attack)
- ◆ WinSxS (Side-by-Side) Assembly
 - ◆ Flexibility to update the binaries
 - ◆ May grant trusted installer privileges

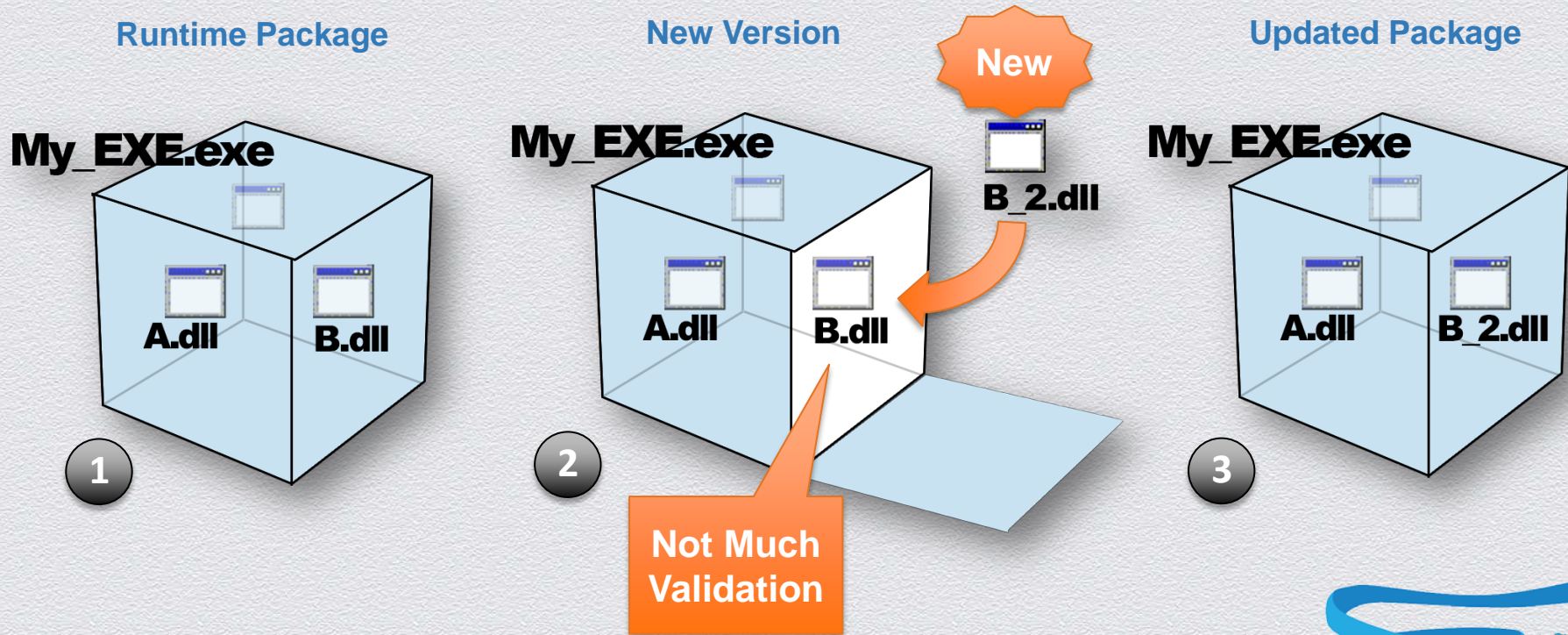
Traditional Search Order High-Jacking

Search for DLL in the first location then the second



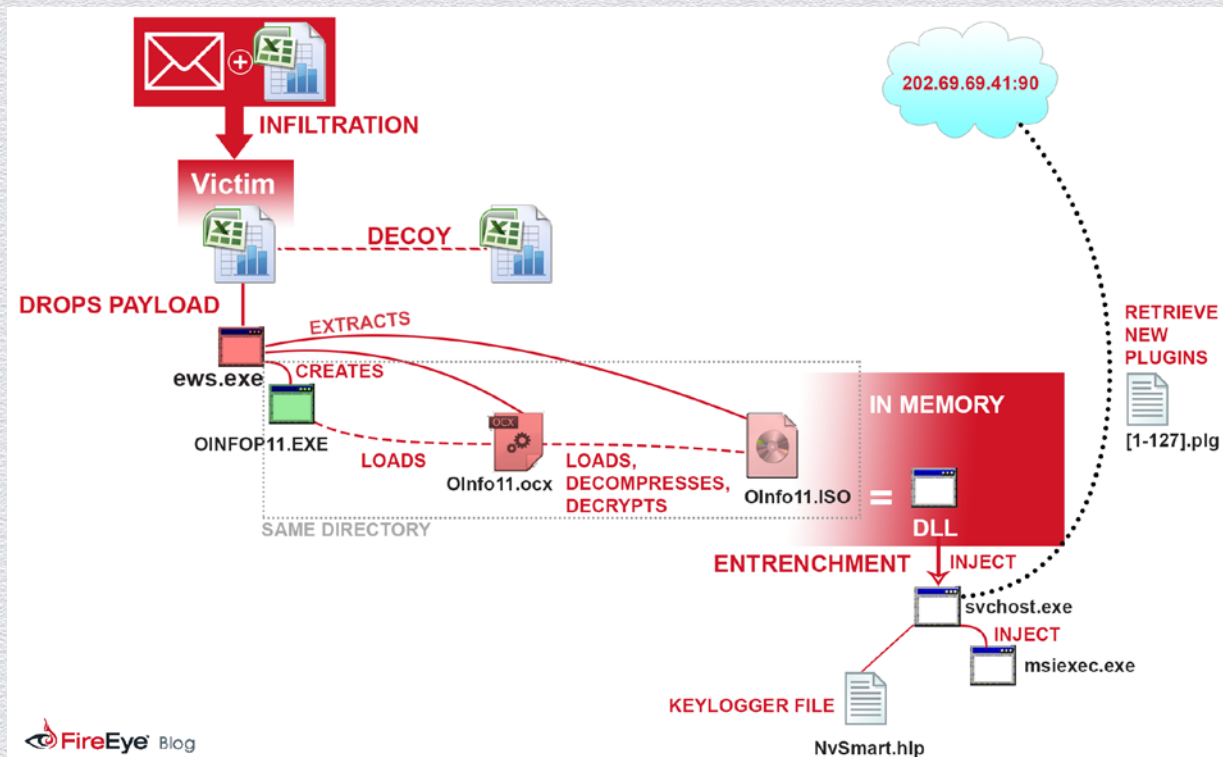
DLL Side-loading

Search for DLL within the executable run path



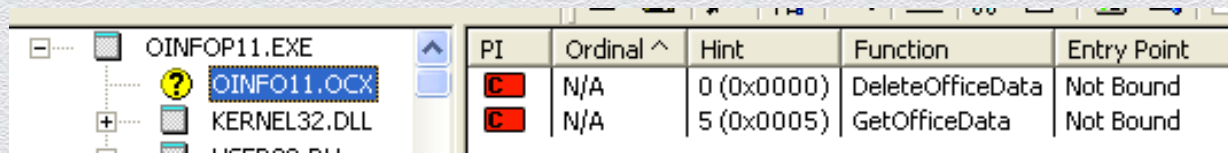
APT PlugX

- ◆ Used in Office 2003 service pack 2 update executable
- ◆ Self Extracting Rar executable drops all 3 files
- ◆ Oinfo11.ocx and Oinfo.ISO same directory



APT PlugX

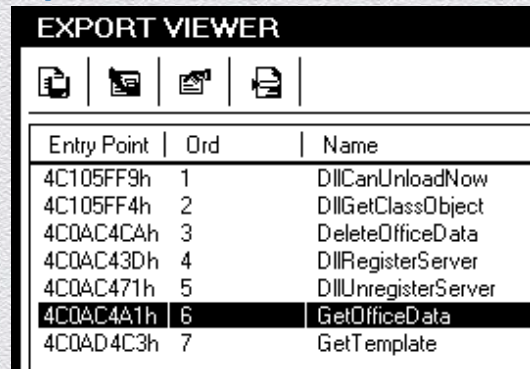
Dependency Walker



A screenshot of the Dependency Walker application. The left pane shows a tree view with 'OINFOP11.EXE' selected, and below it, 'OINFO11.OCX' (marked with a yellow question mark) and 'KERNEL32.DLL'. The right pane displays a table of dependencies.

PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	DeleteOfficeData	Not Bound
C	N/A	5 (0x0005)	GetOfficeData	Not Bound

PE Explorer



A screenshot of the PE Explorer 'EXPORT VIEWER' window. It shows a list of exported functions from a DLL. An orange arrow points from the 'GetOfficeData' entry in this list to the 'GetOfficeData' entry in the Dependency Walker table above.

Entry Point	Ord	Name
4C105FF9h	1	DllCanUnloadNow
4C105FF4h	2	DllGetClassObject
4C0AC4CAh	3	DeleteOfficeData
4C0AC43Dh	4	DllRegisterServer
4C0AC471h	5	DllUnregisterServer
4C0AC4A1h	6	GetOfficeData
4C0AD4C3h	7	GetTemplate

◆ Check for DLL Imports

◆ OINFOP11.EXE

GetOfficeData

OInfo11.ocx

Original Exported Function

```
.text:4C0AC4A1 ; int __cdecl GetOfficeData(int, int lpUserNameBuf, LPCSTR lpOrgNameBuf)
.text:4C0AC4A1 public GetOfficeData
.text:4C0AC4A1 GetOfficeData proc near ; DATA XREF: .text:off_4C131938↓o
.text:4C0AC4A1
.text:4C0AC4A1 arg_0 = dword ptr 4
.text:4C0AC4A1 lpUserNameBuf = dword ptr 8
.text:4C0AC4A1 lpOrgNameBuf = dword ptr 0Ch
.text:4C0AC4A1
.text:4C0AC4A1 push [esp+arg_0]
.text:4C0AC4A5 push offset aGetOfficeDataD ; "GetOfficeData (%d)"
.text:4C0AC4AA call nullsub_1
.text:4C0AC4AF pop ecx
.text:4C0AC4B0 pop ecx
.text:4C0AC4B1 push 0 ; int
.text:4C0AC4B3 push [esp+4+lpOrgNameBuf] ; lpOrgNameBuf
.text:4C0AC4B7 mov ecx, offset unk_4C133848
.text:4C0AC4BC push [esp+8+lpUserNameBuf] ; lpUserNameBuf
.text:4C0AC4C0 push [esp+0Ch+arg_0] ; int
.text:4C0AC4C4 call sub_4C0C8407
.text:4C0AC4C9 retn
.text:4C0AC4C9 GetOfficeData endp
```

Fake Exported Function

```
.text:10001000
.text:10001000 public GetOfficeData
.text:10001000 GetOfficeData proc near ; DATA XREF: .rdata:off_10009B68↓o
.text:10001000 retn ; DeleteOfficeData
.text:10001000 GetOfficeData endp
.text:10001000
.text:10001000 ; -----
```

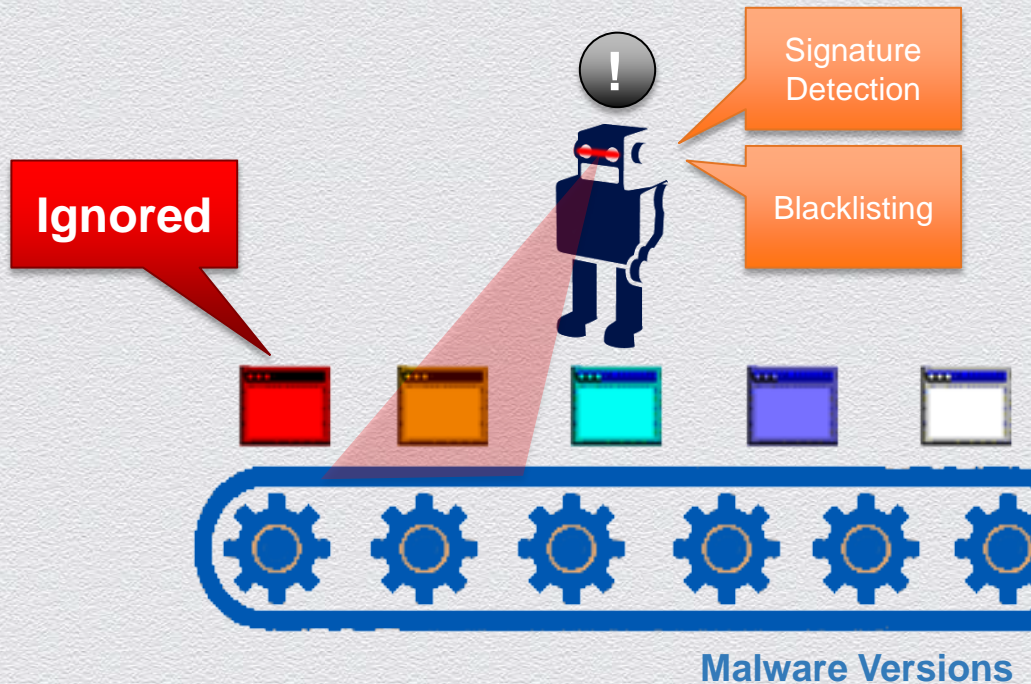

Other Examples

- ◆ mcvsmmap.exe 4e1e0b8b0673937415599bf2f24c44ad
- ◆ NvSmartMaxapp.dll 09b8b54f78a10c435cd319070aa13c28
- ◆ RASTLS.EXE 62944e26b36b1dcace429ae26ba66164

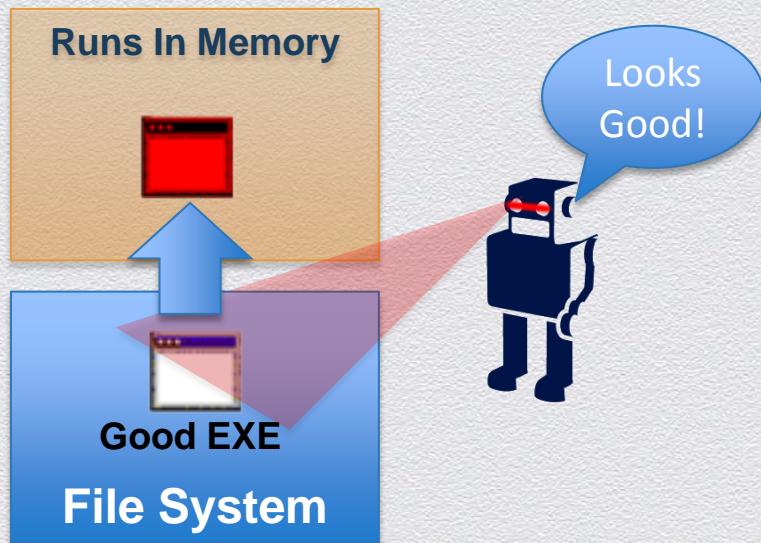
Why is this A Problem?

- ◆ **Slipping by AV Scanners**

- ◆ For every new generated and compressed executable, blacklisting by hash and signature generation will not always be detected
- ◆ OINFOP11.EXE was listed on the NSRL database as a clean file
- ◆ Valid Certs/Binary Signatures



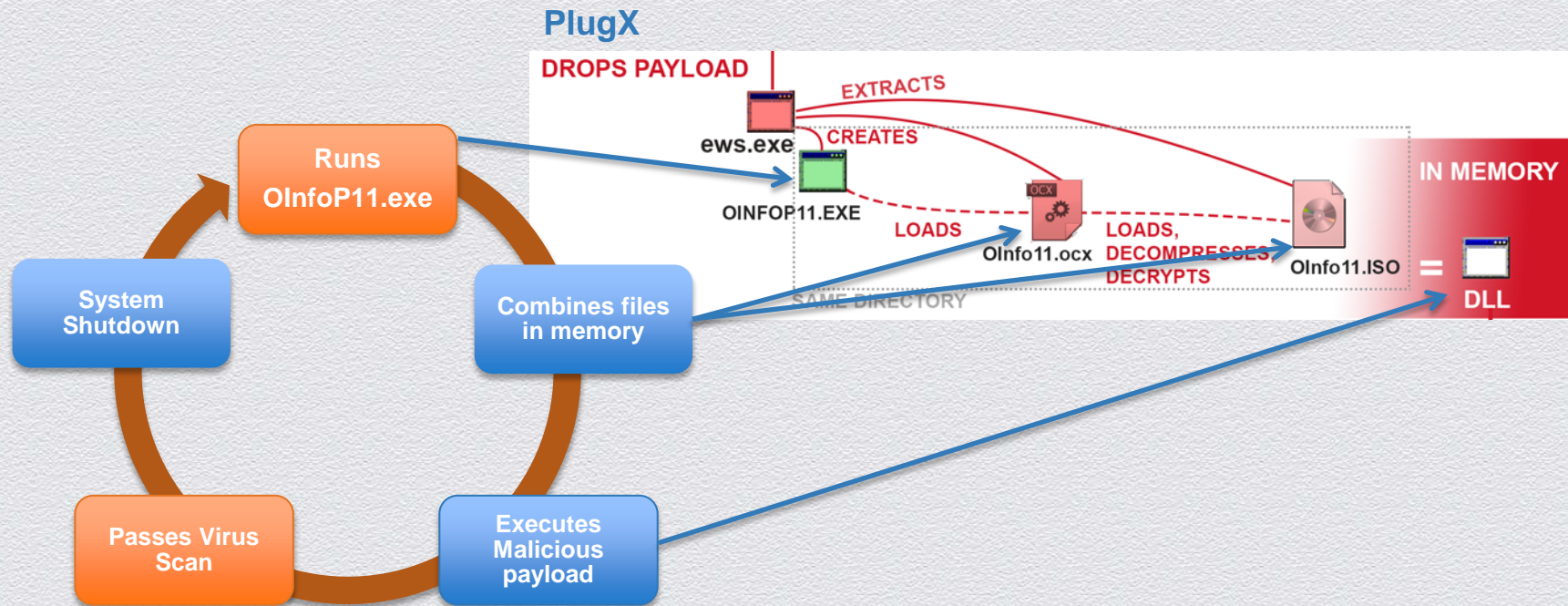
Why is this A Problem?



◆ Persistence




- ◆ If the malicious executable is built in memory, AV scanners cannot detect the catalyst.
- ◆ Every time that executable is called it will always rebuild the malicious executable components

Why is this A Problem?



How to recognize it?

- ◆ Various Tools
 - ◆ PE explorers to view DLL imports
 - ◆ <http://www.heaventools.com/overview.htm>
 - ◆ Dependency Walker
 - ◆ <http://www.dependencywalker.com/>
 - ◆ SxStrace.exe (Found in MS Vista)
 - ◆ Validate manifests and dll tracing

EXPORT VIEWER		
		
Entry Point	Ord	Name
4C105FF9h	1	DllCanUnloadNow
4C105FF4h	2	DllGetClassObject
4C0AC4CAh	3	DeleteOfficeData
4C0AC43Dh	4	DllRegisterServer
4C0AC471h	5	DllUnregisterServer
4C0AC4A1h	6	GetOfficeData
4C0AD4C3h	7	GetTemplate

How to avoid it?

◆ Developer

- ◆ Validate Imported Functions
- ◆ Utilize DLL redirection or a Manifest
- ◆ Call SetDllDirectory with an empty string.

◆ Endpoint User

- ◆ Restrict write and execute permissions user folders
- ◆ Least privilege access

Manifest File Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
manifestVersion="1.0">
  <assemblyIdentity publicKeyToken="75e377300ab7b886"
type="win32" name="Test4Dir" version="1.0.0.0"
processorArchitecture="x86"/>
  <file name="DirComp.dll"
hash="35ca6f27b11ed948ac6e50b75566355f0991d5d9"
hashalg="SHA1">
    <comClass clsid="{6C6CC20E-0F85-49C0-A14D-
D09102BD7CDC}" progid="DirComp.PathInfo"
threadingModel="apartment"/>
    <typelib tid="{AA56D6B8-9ADB-415D-9E10-
16DD68447319}" version="1.0" helpdir=""/>
  </file>
</assembly>
```


RSA[®]CONFERENCE2014

FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO



Questions?