

Automated Detection of COM Vulnerabilities

Frederic Bret-Mounet, CISSP

July 30, 2003

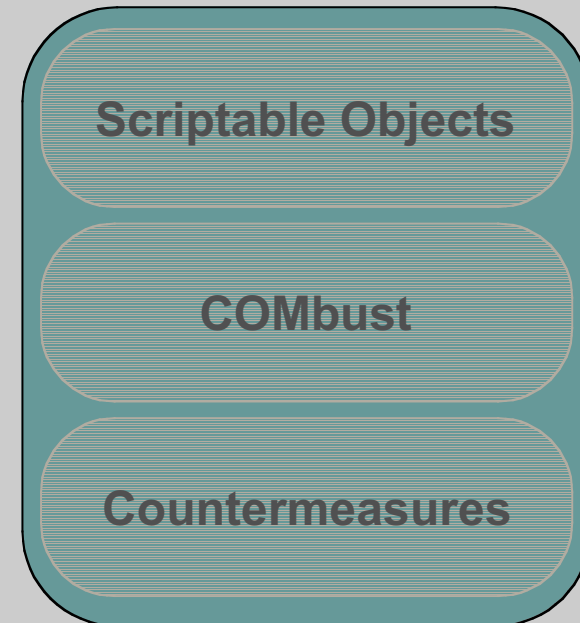
@stake



Where Security & Business IntersectSM

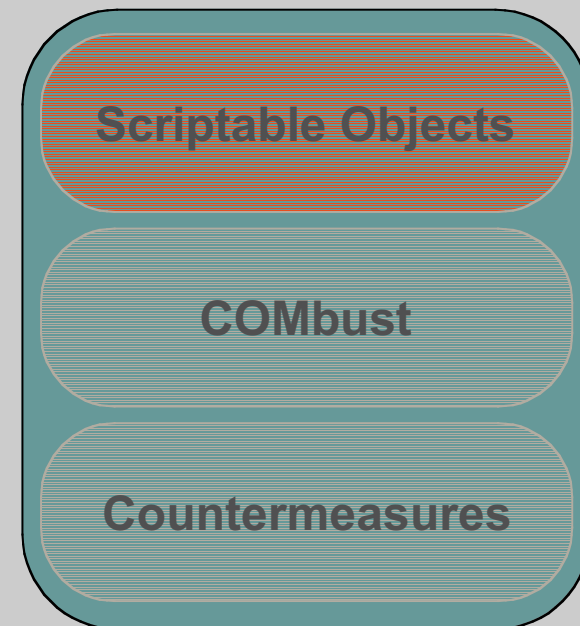
Agenda

- **Scriptable Objects**
- **COMbust – the tool**
- **Countermeasures – Who's calling me?**
- **Q&A**



Scriptable Objects

- What's a scriptable object?
- New Class of Attack
- Attack Scenarios
- Attack Demo



Scriptable Objects

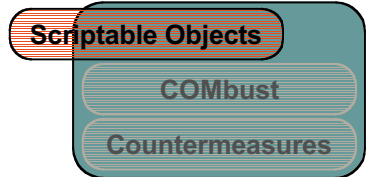
COMbus

Countermeasures

What's a scriptable object?

- In COM speak, an object implementing the IDispatch interface.
- From a user perspective, an object any scripting language can instantiate and interact with.
- Famous automation objects used by the Melissa virus:

```
Set UngaDasOutlook = CreateObject("Outlook.Application")
Set DasMapiName = UngaDasOutlook.GetNameSpace("MAPI")
For y = 1 To DasMapName.AddressLists.Count
    [...]
Next y
```



New Class of Attack

- **Melissa uses the Outlook automation objects as they were intended to be used.**
- **What happens when this functionality is abused?**
 - Buffer overflows
 - Crashes
 - Privilege escalation : user credentials, file access, registry access,...
 - DOS: CPU utilization, hard drive, network bandwidth,...
- **Use an automation object not for its functionality, but as an attack vector.**

Scriptable Objects
 COMbust
 Countermeasures

New Class of Attack (cont...)

	OS	Win XP Pro	Win 2003
	Age	6 month	1 month
	Cruft Force	6 / Limping	2 / Comfortable
COM objects registered		≈ 6000	≈ 4200
COMbustible (IDispatch interface)		≈ 2200 (37%)	≈ 1600 (38%)

➤ Quite a few COMbustion opportunities !

Cruft Scale: <http://www.ddj.com/documents/s=7453/ddj0208q/0208q.htm>

Scriptable Objects

COMbus

Countermeasures

Attack Scenarios

■ Remote:

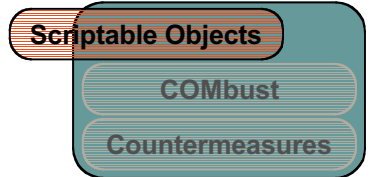
- Internet
- Email
- Intranet
- Newsgroups

■ Local:

- Cross-session on Terminal Services or XP
- Privilege escalation with out-of-proc objects
- Scripts (login, shared scripts,...)

■ Any channel that can use automation:

- VB/JavaScript
- C/C++
- WSH
- Java
- VBA
- Perl / Python
- VB
- Other COM enabled languages

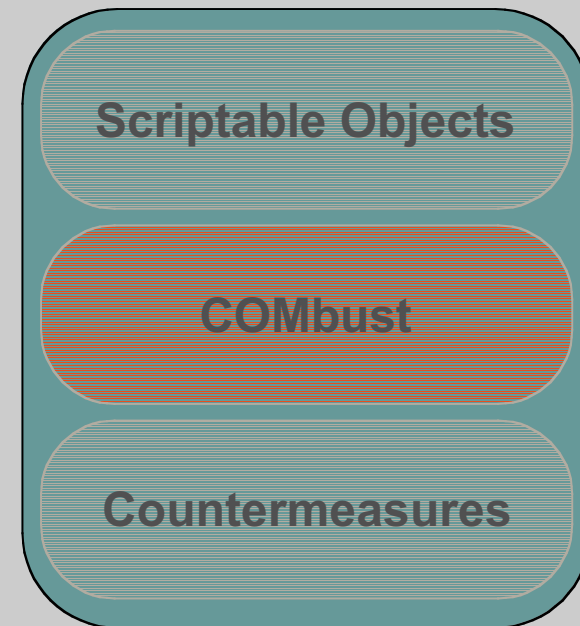


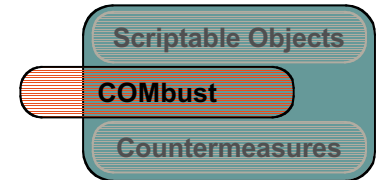
Attack Demo

- **Client-side JavaScript embedded in HTML**
- **HTML delivered from a web site**
- **Victim already has vulnerable object**
 - If not, can also be downloaded
- **Assumes ActiveX execution permission**
 - Is the case of IE out of the box. This activeX was already downloaded by user.
 - If a warning appears, it's often ignored by users

COMbust

- **What does COMbust do?**
- **Configuration and Fuzz Strings**
- **Output**
- **Shortcomings**
- **Advanced Usage**
- **Possible Enhancements**
- **Demo**

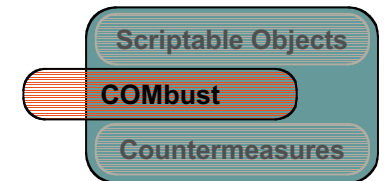




COMbust – What does it do?

- **Exercise a function in boundary cases.**
- **Based on argument types**
- **For example:**
 - Function Foo(string str, boolean b)
 - COMbust will call:
 - Foo(“”,true)
 - Foo(“”,false)
 - Foo(“very long string”,true)
 - Foo(“very long string”,false)
 - Foo(“file path”,true)
 - Foo(“File path”,false)

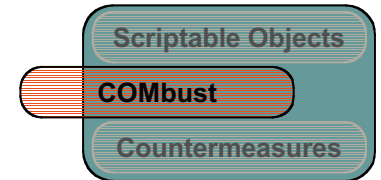
COMbust - Configuration and Fuzz Strings



- **Command line:**

- COMbust -p ProgId | -c CLSID [-f FunctionName]
 - ProgId: eg "msxml2.domdocument.3.0".
 - CLSID: eg "{EF99BD32-C1FB-11D2-892F-0090271D4F88}"
 - -f MyFunction if you only want to fuzz a specific method.
- Either ProgId and CLSID are equivalent, sometimes ProgId not defined.

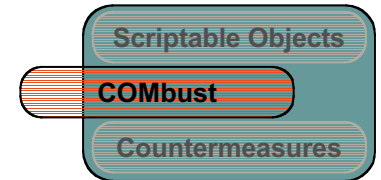
COMbust - Configuration and Fuzz Strings (cont..)



- **Config File:**
 - XML document
 - Same name as executable, in current directory

- **3 sections:**
 - Application configuration
 - Fuzz values for each argument type
 - List of methods to ignore.

COMbust - Configuration and Fuzz Strings (cont..)



- **Application configuration:**

- <MaxCombinations>
 - Maximum number of configurations to be tested per method.
 - Some methods have up to 30 variant argument: $4^{30} = 10^{51}$!
- In the future, more configuration elements such as output format, recursion,...

```
<Config>  
<!-- Max number of iteration a specific function will go through -->  
    <MaxCombinations>1000</MaxCombinations>  
</Config>
```

Scriptable Objects

COMbust

Countermeasures

COMbust - Configuration and Fuzz Strings (cont..)

- **Fuzz values:**

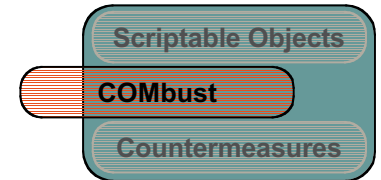
- 1 group per variant type.
- Currently cannot be extended to support more variant types

- **Few modifiers:**

- dt:dt="[...]" to help format conversion to appropriate variant type.
- Count="[n]" to repeat a string pattern n times.

```
<VT_BSTR>
  <Value Label="Path" dt:dt="string">\\?\c:\fuzz.log</Value>
  <value Label="Long string" count="4000" dt:dt="string">A</value>
  <value Label="format string" dt:dt="string">%s %s</value>
</VT_BSTR>
<VT_I1>
  <Value Label="0" dt:dt="i1">0</Value>
  <Value Label="Max" dt:dt="i1">127</Value>
</VT_I1>
```

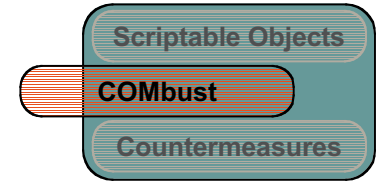
COMbust - Configuration and Fuzz Strings (cont..)



■ Ignore section:

- Methods you want to bypass.
 - For example, AddRef & Release should not be fuzzed as they are core COM functions
 - Methods you want to bypass as they crash or end logical functionality (eg. Close())
- Values can take 2 forms:
 - FunctionName - ignore function with that name regardless of class
 - ProgId::FunctionName – for function of a specific object

```
<Ignore>  
    <value>AddRef</value>  
    <value>MyProgId::MyMethod</value>  
    <value>FooBar</value>  
</Ignore>
```



COMbust – Output

■ Header

ProgId if known - CLSID

COMbust - ver 0, 6, 6, 6 (C) @stake, 2003
 msxml2.domdocument.3.0 - {F5078F32-C551-11D3-89B9-0000F81FE221}

COM Interface for:msxml2.domdocument.3.0

Implemented Interface

Name:IXMLDOMDocument2

Description & Help from TypeInfo

Description:(null)

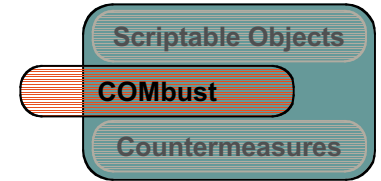
Help File:(null)

Number of Functions and Variables enumerated

Functions:82

Variables:0

Implementations:1



COMbust – Output (cont...)

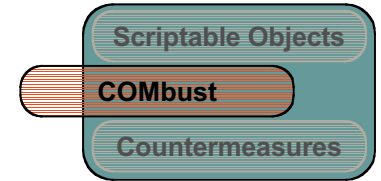
- Method enumeration

For each function:
 -Type & argument signature
 -Help string if available

```

*****
Dispatch Function VOID QueryInterface(PTR riid, void **ppv)
[...]
Dispatch Put VOID ontransformnode(VARIANT ) }
    register an ontransformnode event handler
Dispatch Get PTR namespaces()
    A collection of all namespaces for this document
[...]
Dispatch Function VOID SetProperty(BSTR name, VARIANT value )
    set the value of the named property
Dispatch Function VARIANT GetProperty(BSTR name )
    get the value of the named property
Combinations to be tested: 1985
*****
    
```

Total number of combinations to be tested.



COMbust – Output (cont...)

- Functional fuzzing

IGNORING: Dispatch Function VOID Invoke(I4 dispidMember ,PTR pdispparams ,PTR pvarResult ,PTR pexcepinfo ,PTR puArg

Dispatch Get BSTR nodeName() = "#document"

Dispatch Put VOID nodeValue("")

Dispatch Put VOID nodeValue("AAAAAAAAAAAAAAAAAAAA(3k)")

Dispatch Put VOID nodeValue("BBBBBBBBBBBBBBBBBB(9k)")

[...]

Dispatch Put VOID nodeValue(-99999999.9999)

Dispatch Put VOID nodeValue(99999999.9999)

Dispatch Get USERDEFINED nodeType() = 9

Dispatch Get PTR parentNode() = VT_DISPATCH

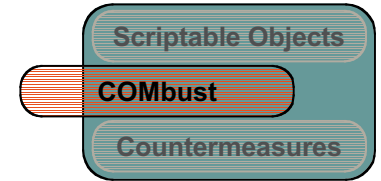
Dispatch Function PTR selectNodes("%s %s") = EXCEPTION:Unexpected character in query string.

-->%<--s %s

For each function:
-Call with all combinations of variant type.

Return value if any

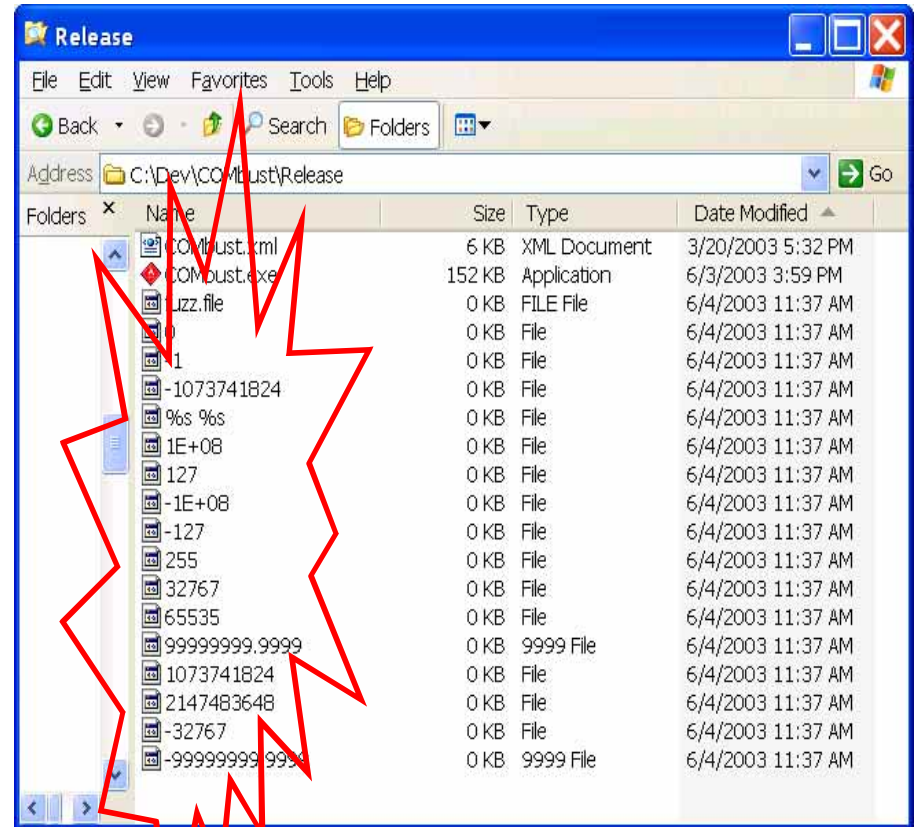
Handled Exception if any



COMbus – Output (cont...)

■ COMbus crashes

- Probably a Buffer Overflow or a DOS potential
- In few cases COMbus bug...
- Can take several forms:
 - Uncaught Exception dialog
 - “CRASH!!!!” in output window
 - Output truncation
- Other things to look for:
 - Files being created / overwritten
 - Registry entries being modified
 - Security tokens



Scriptable Objects

COMbust

Countermeasures

COMbust - Shortcomings

- **Object limits:**
 - Only IDispatch based objects
 - No objects expecting caller interfaces
- **Variants:**
 - Not all variants currently supported:
 - User defined, arrays, pointers
 - Strings limited to ~19700 characters
- **Application:**
 - Probably still has a few bugs ☺ - Double check its findings



Scriptable Objects

COMbust

Countermeasures

COMbust – Words of Wisdom

- **Warning from the Surgeon General:**
 - “Execute at your own risk!”
- **COMbusted Objects have been known to:**
 - Create / overwrite (important) files
 - Put garbage in your registry
 - Render some apps un-launchable
- **Best environment for COMbustion is in a Virtual Machine.**



Scriptable Objects

COMbust

Countermeasures

COMbust - Advanced Usage

■ Scripting Capabilities

- Use `-s` switch on command line.
- `.vbs` extension loads VBScript engine
- `.js` loads JScript engine



Scriptable Objects

COMbust

Countermeasures

COMbust - Advanced Usage (cont...)

- **3 Entry points. Implement the following:**

- NewObject() to create a new object.

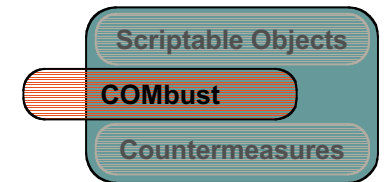
```
function NewObject()  
    set a = CreateObject("My.ProgId.1")  
    set NewObject = a.GetChildObject()  
end function
```

- Finalize(obj) to cleanup an object.

```
Function Finalize(obj)  
    obj.Save()  
end function
```

- Initialize(obj) to initialize an object COMbust created.

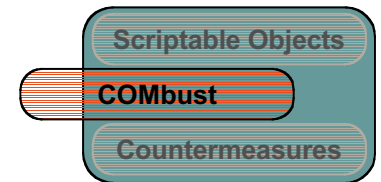
```
Function Initialize(obj)  
    obj.Load("test.xml")  
end function
```



COMbust – Possible Enhancements

- **More Variant types**
- **Recursion**
- **GUI a la OleView**
- **Output format – text, html, csv, xml,...**
- **More scripting support – scripted variants, scriptable COMbust object**
- **Support for IUnknown**
- **Auto script / html generation**
- **Your suggestions are welcome**

COMbust - Demo

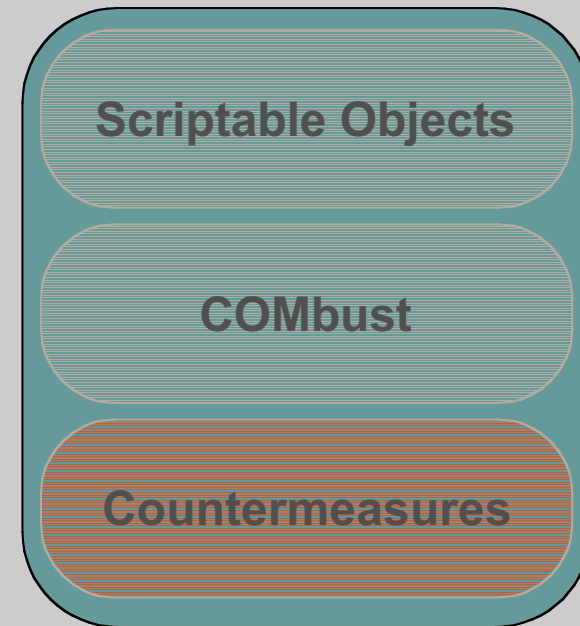


- **Let's have fun!**

- **Demo of:**
 - COMbustion of simple object.
 - COMbustion of vulnerable object.
 - Advanced COMbustion of non creatable child object.

Countermeasures

- **Don't use IDispatch if you can!**
- **Software Best Practices**
- **Otherwise a few options to explore:**
 - SiteLock
 - Expected Host Interfaces
 - Crypto



Scriptable Objects

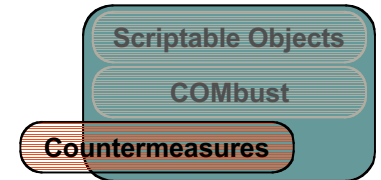
COMbust

Countermeasures

Countermeasures – Software Best Practices

- **Input validation, Input validation and Input validation!**
 - Keep in mind this is untrusted input.
- **Code review**
 - Should be performed from a functional standpoint, but also from a security perspective.
- **Security Awareness**
 - Formal or informal training, knowledge sharing,...
- **Unit test**
 - Often overlooked.
- **Run COMbust!**
 - Build a set of automated tests for unit and regression testing.

Countermeasures – Other Options



- **For ActiveX controls:**
 - Implement SiteLock Template
- **Enforce a contractual obligation on the hosts' part.**
 - Expect callback interface or event sinks.
- **Use PKI or Crypto to authenticate caller.**
- **Sign code.**
 - No, not really!
- **Build Defense in Depth.**

Scriptable Objects

COMbust

Countermeasures

Credits

- **Nathalie, Laurent, Vanessa and Raphael for letting me play way past bedtime...**
- **My colleagues Adrian, Alex, Cyclips, Lefty and Weld Pond for their suggestions, being guinea pigs and support**
- **Most importantly, a hotel room for imprisoning me and freeing my ideas!**

Scriptable Objects

COMbust

Countermeasures

Q & A

- fbret@stake.com

- **References:**

- COMbust Download: <http://www.atstake.com/research/tools/>
- ActiveX security:
<http://msdn.microsoft.com/workshop/components/activex/security.asp>
- SiteLock:
<http://msdn.microsoft.com/downloads/samples/internet/components/SiteLock/default.asp>
- ActiveX signing: http://msdn.microsoft.com/library/en-us/dnaxctrl/html/msdn_signmark.asp