Measuring Relative Attack Surfaces

Jeannette Wing

School of Computer Science
Carnegie Mellon University

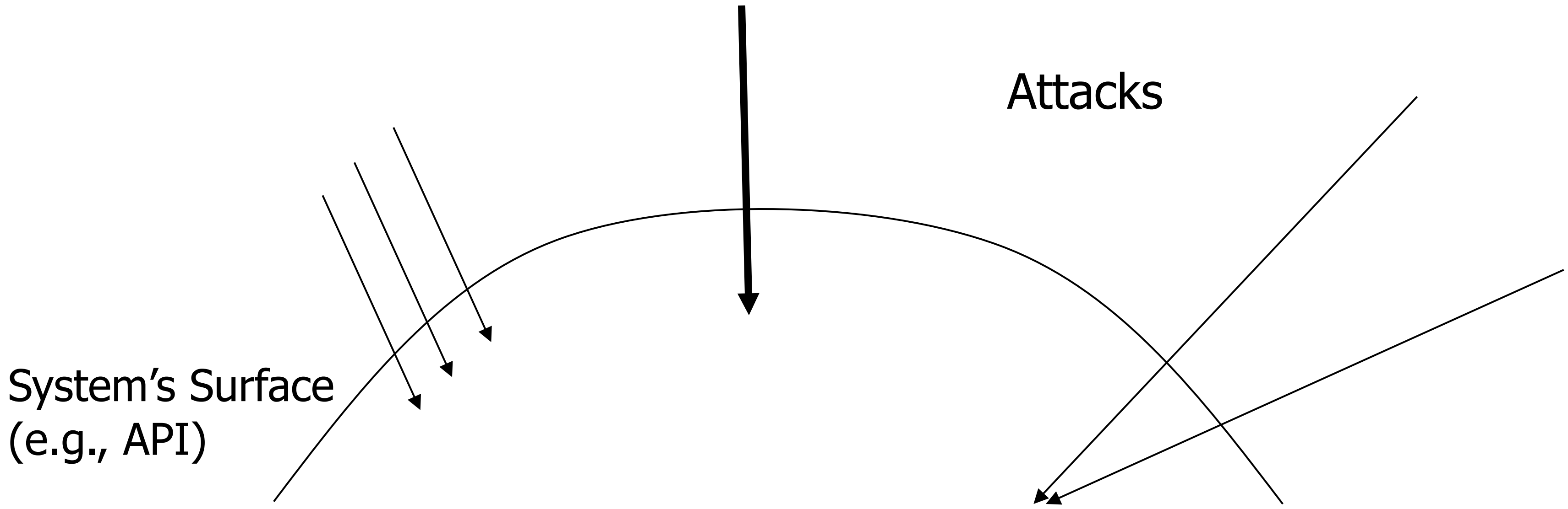
Joint with Mike Howard and Jon Pincus, Microsoft Corporation

Motivation

- How do we measure progress?
  - What effect has Microsoft's Trustworthy Computing Initiative had on the security of Windows? Has it paid off?
  - What metric can we use to say Windows Server 2003 is "more secure" than Windows 2000?

- One approach: Howard's Relative Attack Surface Quotient (RASQ)

Attackability



Intuition

Reduce the ways attackers can penetrate surface

⇨ Increase system's security

Relative Attack Surface

- Intermediate level of abstraction
  - Impartial to numbers or types of code-level bugs, e.g., #buffer overruns
  - More meaningful than counts of CVE/MSRC/CERT bulletins and advisories

- Focus on *attack vectors*
  - Identify potential features to attack, based on past exploits

    Features to Attack * Security Bugs = Exploits

  - Fewer features to attack implies fewer exploits

- Focus on *relative* comparisons

20 RASQ Attack Vectors for Windows [Howard03]

- Open sockets
- Open RPC endpoints
- Open named pipes
- Services
- Services running by default
- Services running as SYSTEM
- Active Web handlers
- Active ISAPI Filters
- Dynamic Web pages
- Executable vdirs
- Enabled accounts
- Enabled accounts in admin group
- Null Sessions to pipes and shares
- Guest account enabled
- Weak ACLs in FS
- Weak ACLs in Registry
- Weak ACLs on shares
- *VBScript enabled*
- *Jscript enabled*
- *ActiveX enabled*

Relative Attack Surface Quotient

$$\sum_{v \varepsilon AV} \omega_v \; |V|$$

simplistic count

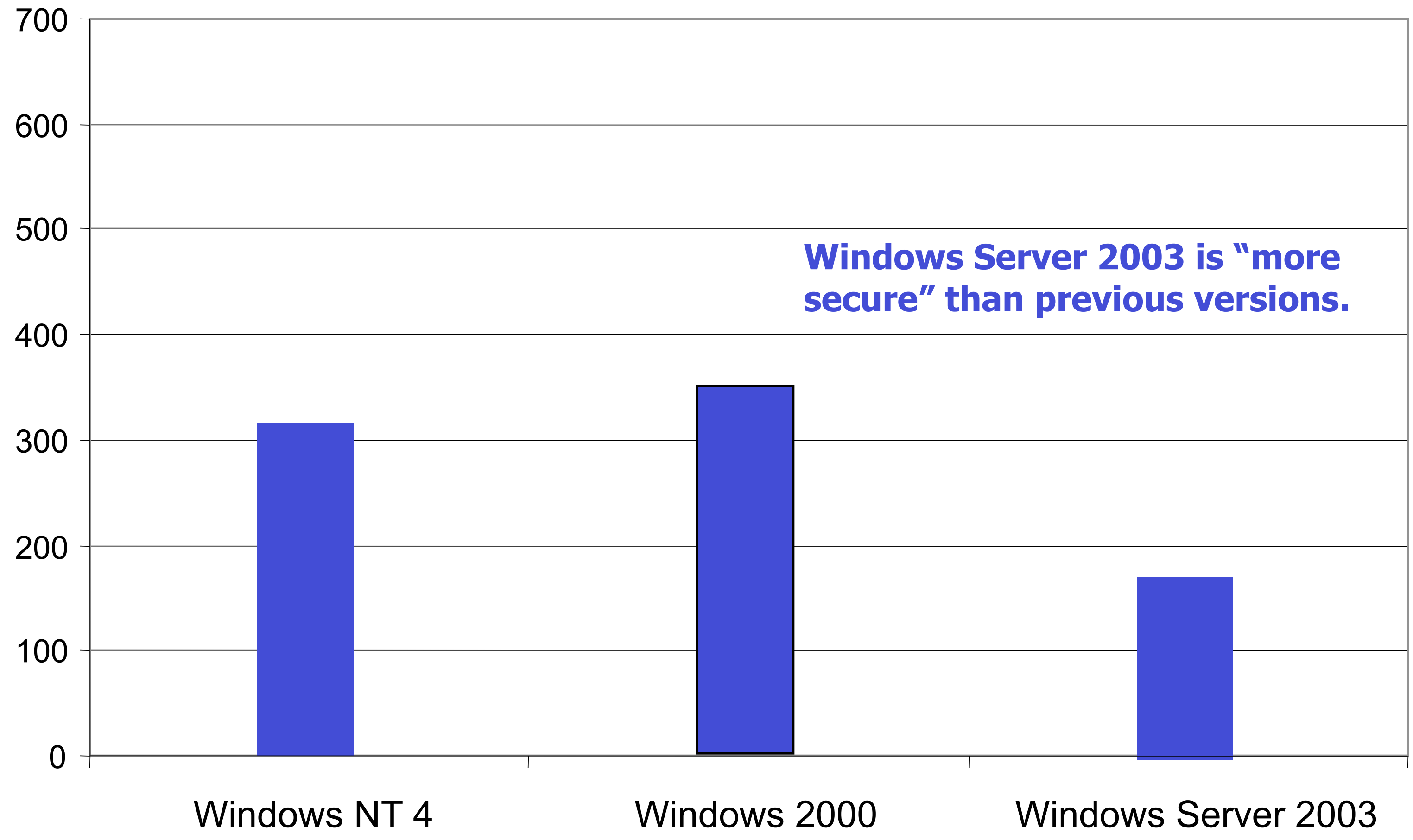
where

v attack vector

ω_v weight for attack vector

AV set of attack vectors

RASQ Computations for Three OS Releases



What's Really Going On?

Informal Definitions

A vulnerability is an error or weakness in design, implementation, or operation.

- "error" => actual behavior – intended behavior

An attack is the means of exploiting a vulnerability.

– "means" => sequence of actions

A threat is an adversary motivated and capable of exploiting a vulnerability.

– "motivated" => GOAL
– "capable" => state entities (processes and data)

[Schneider, editor, *Trust in Cyberspace*, National Academy Press, 1999]

State Machines

$$M = \langle S, I, A, T \rangle$$

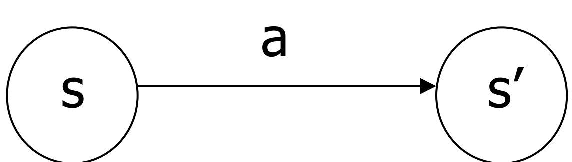
S set of states

$s \in S$, s: Entities $\rightarrow$ Values

$I \subseteq S$ set of initial states

A set of actions

T transition relation



$\langle s, a, s' \rangle \in T$

Execution of action a in state s resulting in state s'

We will use a.pre and a.post for all actions $a \in A$ to specify T.

Behaviors

An execution of M

$$s_0\ a_1\ s_1\ a_2 \ldots\ s_{i-1}\ a_i\ s_i\ \ldots$$

- $s_0 \in I$, $\forall\, i > 0\ \ \langle s_{i-1}, a_i, s_i \rangle \in T$
- infinite or finite, in which case it ends in a state.

The behavior of state machine M, Beh(M), is the set of all its executions.

The set of reachable states, Reach(M), …

System-Under-Attack

$$\text{System} = <S_{sys}, I_{sys}, A_{sys}, T_{sys}>$$
$$\text{Threat} = <S_{thr}, I_{thr}, A_{thr}, T_{thr}>$$

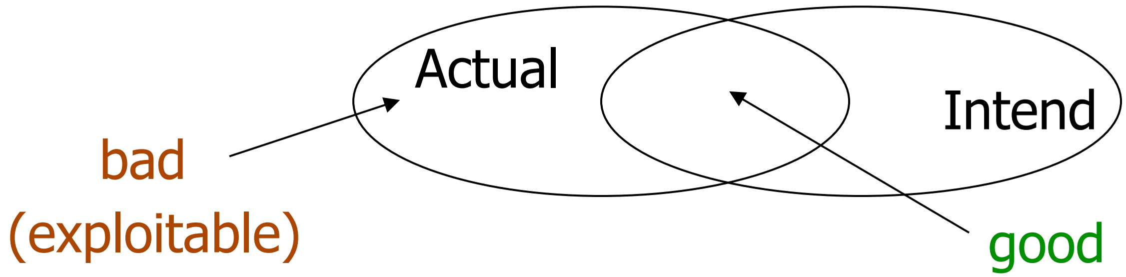
System-Under-Attack = (System || Threat) X GOAL

- || denotes parallel composition of two state machines, interleaving semantics
- GOAL
  - Predicate on state
  - Intuitively, adversary's goal, i.e., "motivation"

Vulnerabilities

$$\text{Actual} = \langle S_{act}, I_{act}, A_{act}, T_{act} \rangle$$
$$\text{Intend} = \langle S_{int}, I_{int}, A_{int}, T_{int} \rangle$$

$$\text{Vul} = \text{Beh}(\text{Actual}) - \text{Beh}(\text{Intend})$$



- $I_{act} - I_{int} \neq \emptyset$
- $T_{act} - T_{int} \neq \emptyset$

  For some action $a \in A_{act} \cap A_{int}$
  - $a_{int}.pre \Rightarrow a_{act}.pre$, or
  - $a_{int}.post \Rightarrow a_{act}.post$

Informally, we'll say "a is a vulnerability."

System-Under-Attack (Revisited)

$$\text{Actual} = <S_{act}, I_{act}, A_{act}, T_{act}>$$
$$\text{Intend} = <S_{int}, I_{int}, A_{int}, T_{int}>$$
$$\text{Threat} = <S_{thr}, I_{thr}, A_{thr}, T_{thr}>$$

Adversary can achieve GOAL:

System-Under-Attack = (Actual || Threat) X GOAL

Adversary cannot achieve GOAL:

System-Under-Attack = (Intend || Threat) X GOAL

Attacks in (Actual || Threat) X GOAL

An attack is a sequence of action executions

$$a_1 \; a_2 \; a_3 \; \ldots \; a_i \; \ldots \; a_n$$

$$s_0 \nearrow \qquad\qquad\qquad \nwarrow s_n$$

such that

- $s_0 \in I$
- GOAL is true in s_n
- There exists $1 \leq i \leq n$ such that a_i is a vulnerability.

Elements of an Attack Surface: State Entities

- Running processes, e.g., browsers, mailers, database servers
- Data resources, e.g., files, directories, registries, access rights
  - carriers
    - extract_payload: carrier -> executable
    - E.g., viruses, worms, Trojan horses, email messages, web pages
  - executables
    - multiple eval functions, eval: executable -> unit
      - applications (Word, Excel, …)
      - browsers (IE, Netscape, …)
      - mailers (Outlook, Oulook Express, Eudora, …)
      - services (Web servers, databases, scripting engines, …)
      - application extensions (Web handlers, add-on dll's, ActiveX controls, ISAPI filters, device drivers, …)
      - helper applications (dynamic web pages, …)

Targets and Enablers

data target → data resource; process target → running process

- Target
  - Any distinguished data resource or running process used or accessed in an attack.
    - "distinguished" is determined by security analyst and is likely to be referred to in Goal.

- Enabler
  - Any state entity used or accessed in an attack that is not a data or process target.

Channels and Protocols

- Channels: means of communication
  - Message passing
    - Senders and receivers
    - E.g., sockets, RPC endpoints, named pipes
  - Shared memory
    - Writers and readers
    - E.g., files, directories, and registries
- Protocols: rules for exchanging information
  - Message passing
    - E.g., ftp, RPC, http, streaming
  - Shared memory
    - E.g., single writer blocks all other readers and writers

Access Rights

$$\text{Access Rights} \subseteq \text{Principals X Objects X Rights}$$

where

Principals = Users ∪ Processes

Objects = Processes ∪ Data

Rights, e.g., {read, write, execute}

- Derived relations
  - accounts, which represent principals
    - special accounts, e.g., guest, admin
  - trust relation or speaks-for relation [LABW92]
    - E.g., ip1 trusts ip2 or Alice speaks-for Bob
  - privilege level
    - E.g., none < user < root

Attack Surface Dimensions: Summary

Channels x Protocols

message passing, shared memory
RPC, streaming, ftp, R/W, …

- server-client web connection C

- MSHTML (process target)
- HTTPD web server W (process enabler)
- Browser B (process enabler)
- HTML document D (carrier, enabler)
- Extracted payload E (executable, enabler)

Targets & Enablers

Processes

Data
- carriers
- executables

- Zone Z

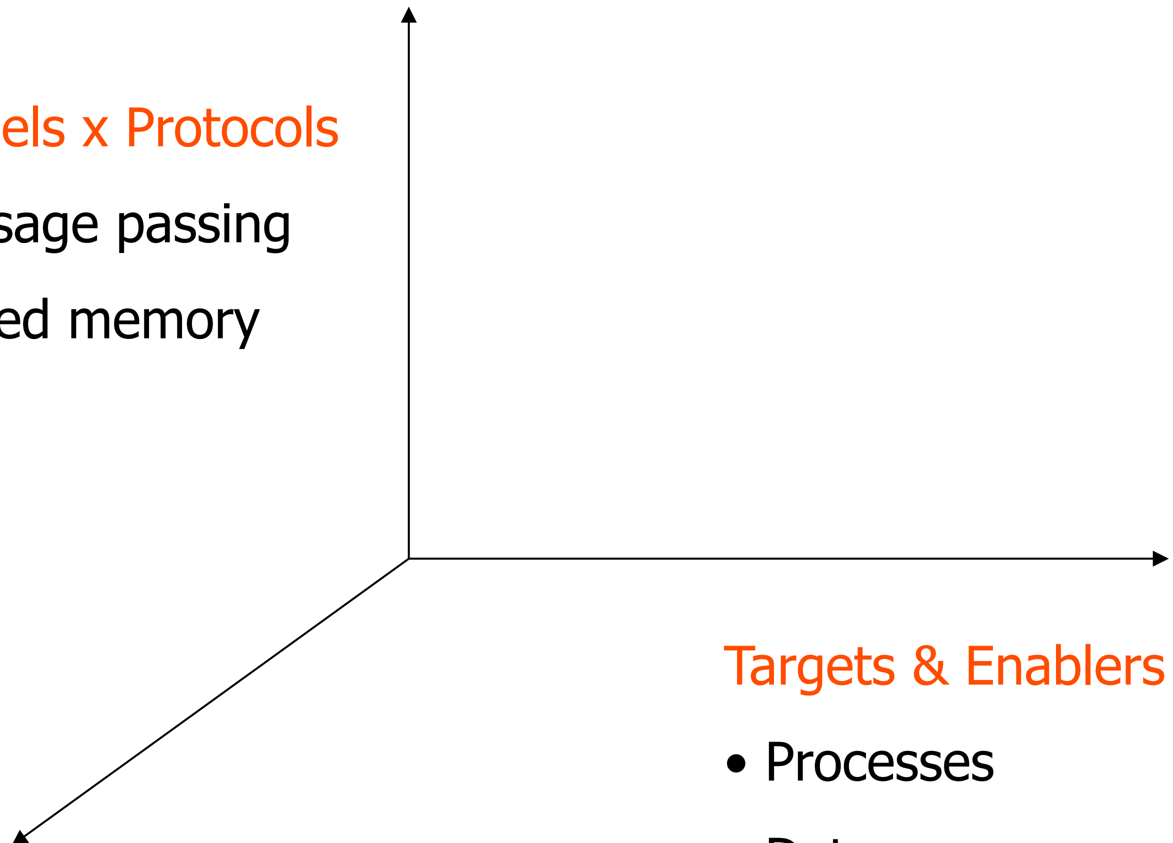
Access Rights

Principals x Objects x Rights

Reducing the Attack Surface

Colloquial		Formal
Turn off macros	→	Eliminate an eval function for one data type.
Block attachments in Outlook	→	Avoid giving any executable as an arg to an eval.
Secure by default	→	Eliminate entire types of targets, enablers, channels; restrict access rights.
Check for buffer overruns	→	Strengthen post-condition of actual to match intended.
Validate your input.	→	Strengthen pre-condition of actual to match intended.
Change your password every 90 days.	→	Increase likelihood that the authentication mechanism's pre-condition is met.

Attack Surface Dimensions: Summary



Channels x Protocols

- message passing
- shared memory

Targets & Enablers

- Processes
- Data
  - carriers
  - executables

Access Rights

Principals x Objects x Rights

Examples

MS02-005

Cumulative Patch for Internet Explorer
(vulnerability 1)

http://www.microsoft.com/technet/security/bulletin/MS02-005.asp

Informally:

- An HTML document (a web page sent back from a server or HTML email) can embed another object using the EMBED tag
- the processing for this tag involves a buffer overrun
- so a well-crafted (valid, but long) tag can lead to arbitrary code execution within the security context of the user.

MS02-005(1): Vulnerability

Action: Action: MSHTML processes HTML document D in zone Z

Intended Precondition: true

Actual Precondition: D contains <EMBED SRC=X> => length(X) <= 512

Intended Postcondition:

    [D contains <EMBED SRC=X> and "Run ActiveX Controls and Plugins" is enabled for Z]
            => display(X)
    // and many other clauses ...

Actual Postcondition (due to non-trivial precondition):

    [D contains <EMBED SRC=X> and "Run ActiveX Controls and Plugins" is enabled for Z]
            => [length(X) > 512 & extract_payload(X) = E] => [E.pre => E.post]
                and [length(X) <= 512] => display(X)
    // and many other clauses ...

MS02-005(1): Web server attack on client

Goal: execute arbitrary code on client via browser

Resource	**Carrier?**	**Channel?**	**Target?**
HTTPD (Web server; process)			
Server-client web connection C		Msg Passing	
Browser (process) B			
HTML document D	Y		
MSHTML (process)			Y

MS02-005(1): Web Server Attack Details

Preconditions (for attack):

- victim requests a web page from adversary site S
- victim has mapped S into zone Z
- victim has "Run ActiveX Controls and Plugins" security option enabled for zone Z
- adversary creates HTML document D with a maliciously-formatted embed tag <EMBED X>, where length(X) > 512 and extract_payload(X) = E

Actions:

1. S sends HTML document D to browser B over connection C
2. B passes D to MSHTML (with zone = Z)
3. **MSHTML processes D in zone Z.**

Postcondition (result of attack): arbitrary effects
(due to post-condition of evaluating E)

MS02-005(1): HTML mail attack

Goal: execute arbitrary code on client via OE

Resource	**Carrier?**	**Channel?**	**Target?**
Mail server S			
Server-client mail connection C		Msg Passing	
Outlook Express (process) OE			
HTML document D	Y		
MSHTML (process)			Y

MS02-005(1): Web Server Attack Details

Preconditions (for attack):

- victim able to receive mail from adversary
- victim receives HTML e-mail in zone Z (where Z != "Restricted Zone")
- victim has "Run ActiveX Controls and Plugins" security option enabled for zone Z
- adversary creates HTML document D with a maliciously-formatted embed tag <EMBED X>, where length(X) > 512 and extract_payload(X) = E

Actions:

1. adversary sends HTML document D to victim via email (via C)
2. victim views (or previews) D in OE
3. OE passes D to MSHTML (with zone = Z)
4. **MSHTML processes D in zone Z.**

Postcondition (result of attack): arbitrary effects
(due to post-condition of evaluating E)

Estimating attack surface, revisited

Measuring the Attack Surface

surface_area = ***f*** (targets, enablers, channels, access rights)

- ***f*** is defined in terms of
  - *relationships* on targets, enablers, channels, …
    - E.g., number of channels per instance of target type.
  - *weights* on targets, enablers, channels, …
    - E.g., to reflect that some targets are more critical than others or that certain instances of channels are less critical than others.

  - Likely to be some function of targets, enablers, channels "subject to" the constraints in access rights.

Mike's Sample Attack Vectors

Channels:
- Open sockets
- Open RPC endpoints
- Open named pipes
- Null Sessions to pipes and shares

Process Targets:
- Services
- Services running by default *
- Services running as SYSTEM *
- Active Web handlers
- Active ISAPI Filters

Data Targets:
- Dynamic Web pages
- Executable vdirs
- Enabled Accounts
- Enabled Accounts in admin group *
- Guest account enabled *
- Weak ACLs in FS *
- Weak ACLs in Registry *
- Weak ACLs on shares *

* = constrained by access rights

Computing RASQ (Mike's model)

$$RASQ = surf_{ch} + surf_{pt} + surf_{dt}$$

where

$surf_{ch}$ = channel surface
$surf_{pt}$ = process target surface
$surf_{dt}$ = data target surface

(each as constrained by access rights)

Computing "channel surface" (Mike's model)

chtypes = { socket, endpoint, namedpipe, nullsession }

$$surf_{ch} = \left[\sum_{c \,\varepsilon\, chtypes} \sum_{i=1}^{|c|} weight(c_i) \right] \Big| A$$

Where

weight(s: socket) = 1
weight(e: endpoint) = 0.9
weight(n: namedpipe) = 0.8
weight(n: nullsession) = 0.9

Computing "process target surface" (Mike's model)

pttypes = { service, webhandler, isapi, dynpage }

$$surf_{pt} = \left[\sum_{p \, \varepsilon \, pttypes} \sum_{i=1}^{|p|} weight(p_i) \right] \Big| A$$

Where

weight(s: service) = 0.4 + default (s) + admin (s)

where default (s) = 0.8 if s = default, 0 otherwise

admin (s) = 0.9 if s = admin, 0 otherwise

weight(w: webhandler) = 1.0

weight(i: isapi) = 1.0

weight(d: dynpage) = 0.6

Computing "data target surface" (Mike's model)

dttypes = { accounts, files, regkeys, shares, vdirs}

$$surf_{dt} = \Big[\sum_{d \,\varepsilon\, dttypes} \; \sum_{i=1}^{|d|} weight(d_i) \Big] \Big|_A$$

Where

weight(a: account) = 0.7 + admin(a) + guest(a)

where admin(a) = 0.9 if a ∈ AdminGroup, 0 otherwise

guest(a) = 0.9 if a.name = "Guest", 0 otherwise.

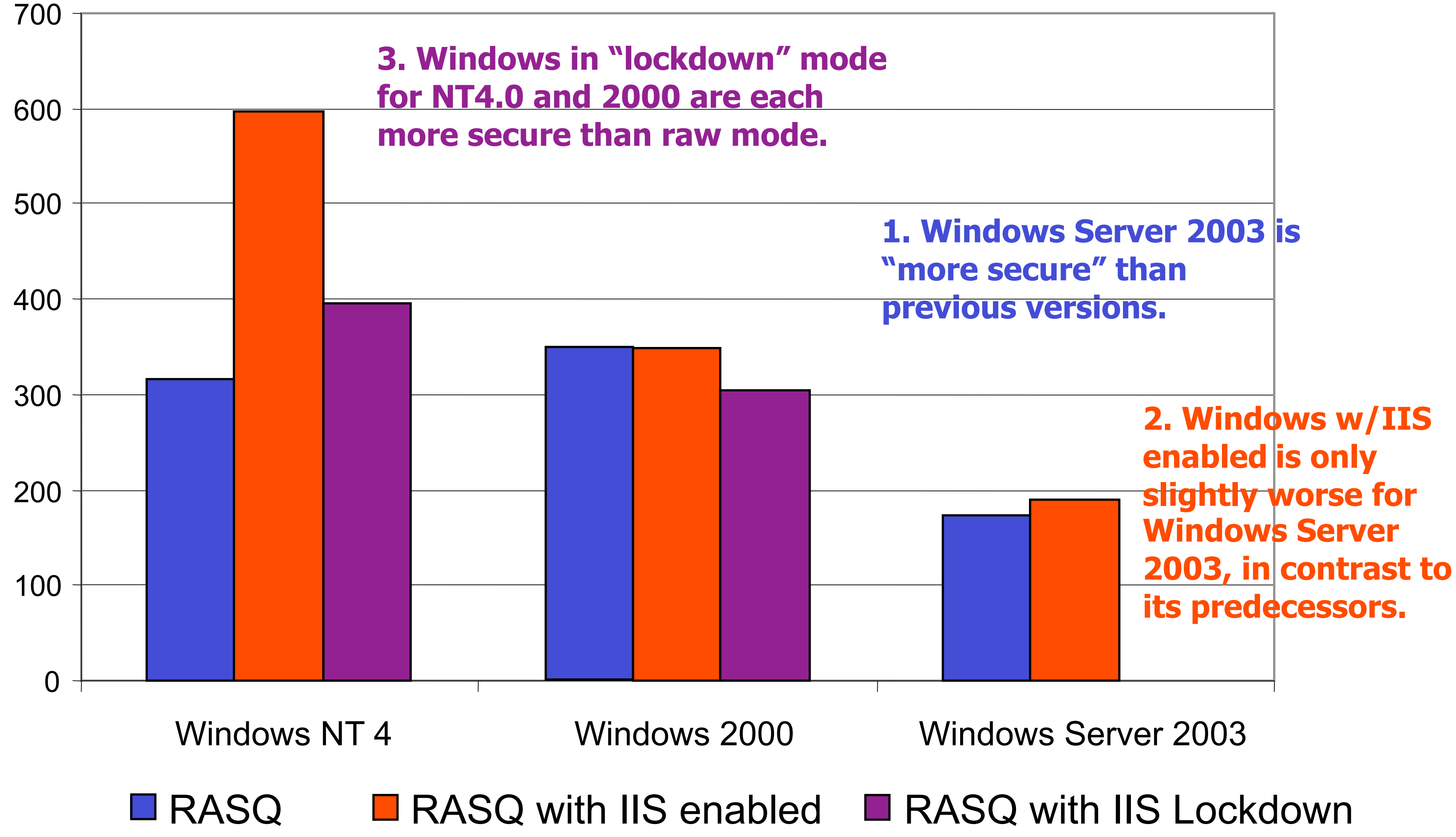
weight(f: file) = 0.7 if weakACL(f), 0 otherwise

weight(r: regkey) = 0.4 if weakACL(r), 0 otherwise

weight(s: share) = 0.9 if weakACL(s), 0 otherwise

weight(v: vdir) = 1.0 if v is executable, 0 otherwise

RASQ Computations for OS Releases



MS02-005a: Cumulative Patch for IE

Attack Sequence:

1. HTTPD web server W sends document D to browser B over connection C.
2. B passes D to MSHTML in zone Z.
3. MSHTML processes D in zone Z, extracting and evaluating E.

Attacker's Goal: Execute arbitrary code E on client

Vulnerability = Actual Behavior – Intended Behavior

Actual Behavior:

D contains <EMBED SRC=X>
∧"Run ActiveX Controls" is enabled for Z
∧ length(X) > 512
=> extract_payload(X) = E and eval(E)

Intended Behavior:

D contains <EMBED SRC=X>
∧ "Run ActiveX Controls" is enabled for Z
=> display(X)

extract_payload: **carrier** → **executable**

eval: **executable** → ()

Caveats

- RASQ numbers are for a given configuration of a running system.
  - They say NOTHING about the inherent "security" of the system after you've turned on the features that were initially off by default!
- It's better to look at numbers for individual attack vector classes rather than read too much into overall RASQ number.
- Mustn't compare apples to oranges.
  - Attack vectors for Linux will be different than those for Windows.
  - Threat models are different.

Short-term technical challenges

- Missing some vectors (ActiveX, enablers like scripting engines, etc.)
  - Approach: analyze MSRC bulletins
- "not all sockets are created equal"
  - Approach: include notion of protocols in RASQ
- Does it really mean anything?
  - Approach: validate with lockdown scenarios, Win2k3 experiences

Research opportunities

- Research on RASQ
  - Measurement aspects: “weights”, combining by adding
  - Applying to things other than the OS
  - Extend to privacy (PASQ?)
  - Finer granularity than “whole system”
    - What things compose?
- Related areas
  - Interactions with threat modeling, attack graphs
  - Identifying opportunities for mitigation
  - Relating to architecture and design principles