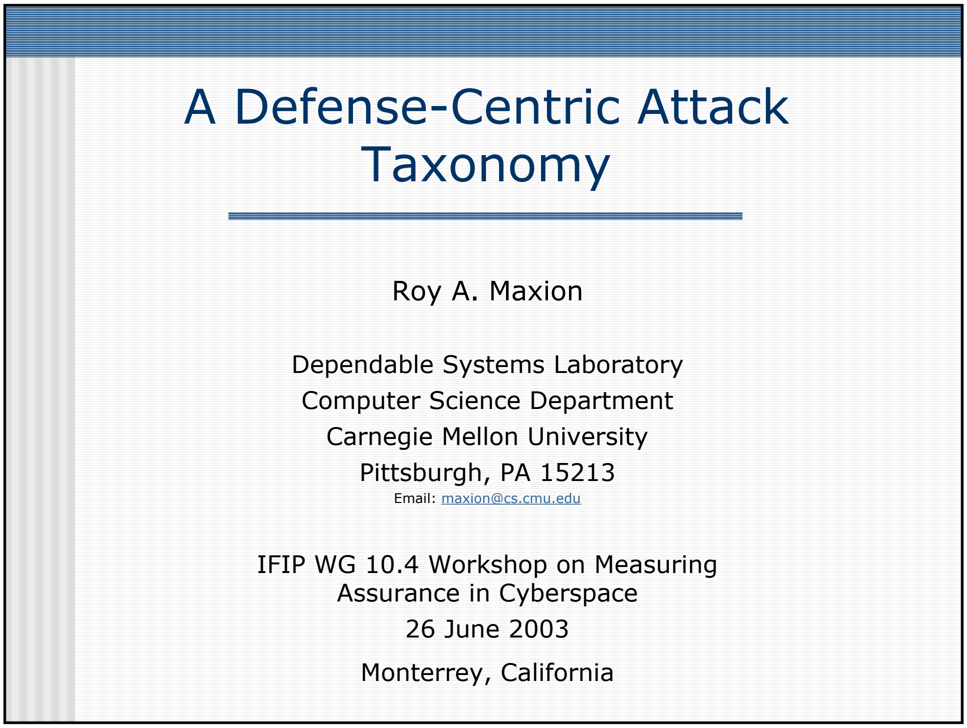
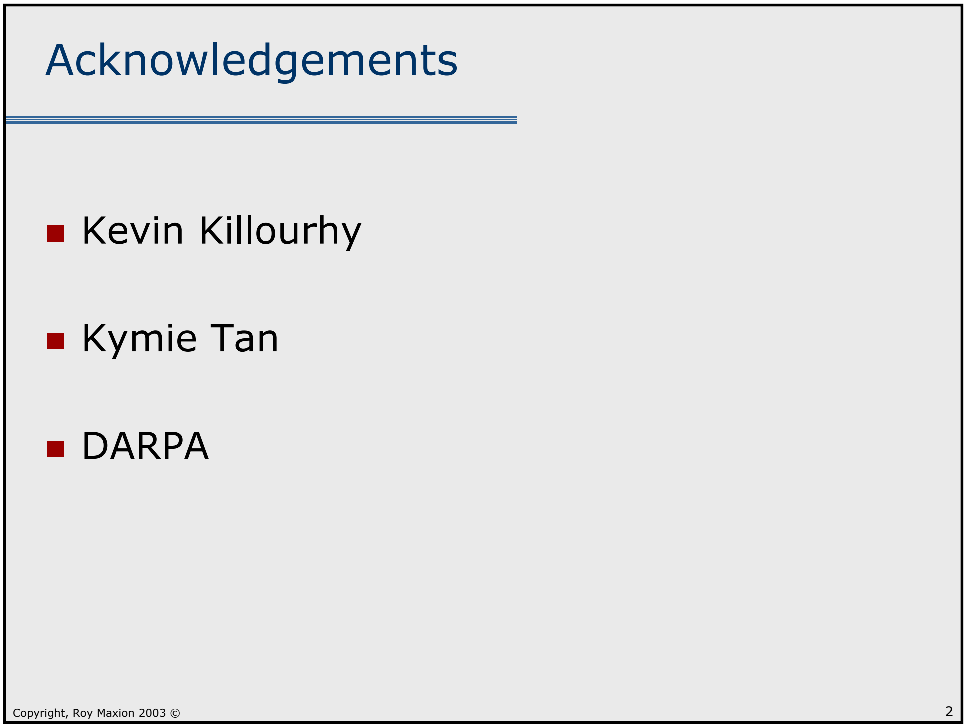


A Defense-Centric Attack Taxonomy

Roy A. Maxion

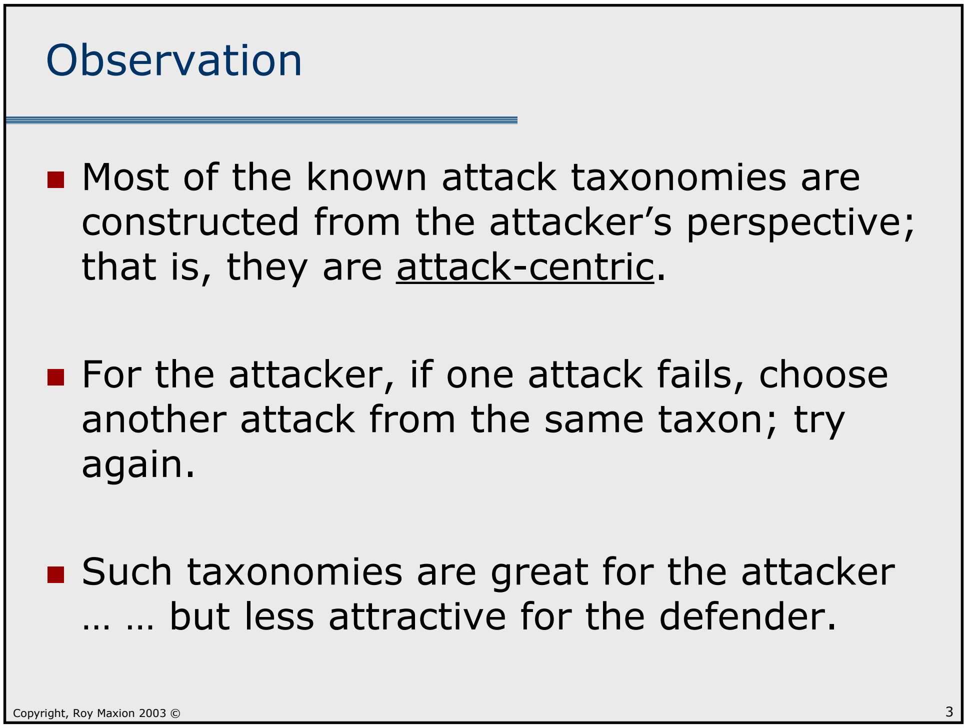
Dependable Systems Laboratory
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
Email: maxion@cs.cmu.edu

IFIP WG 10.4 Workshop on Measuring Assurance in Cyberspace
26 June 2003
Monterrey, California



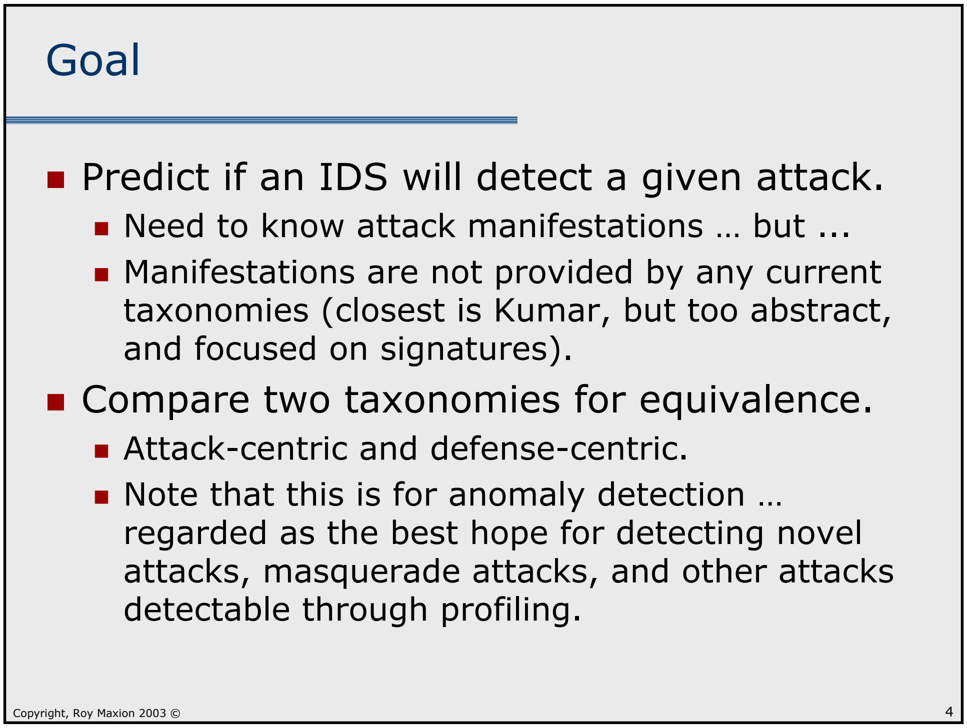
Acknowledgements

- Kevin Killourhy
- Kymie Tan
- DARPA



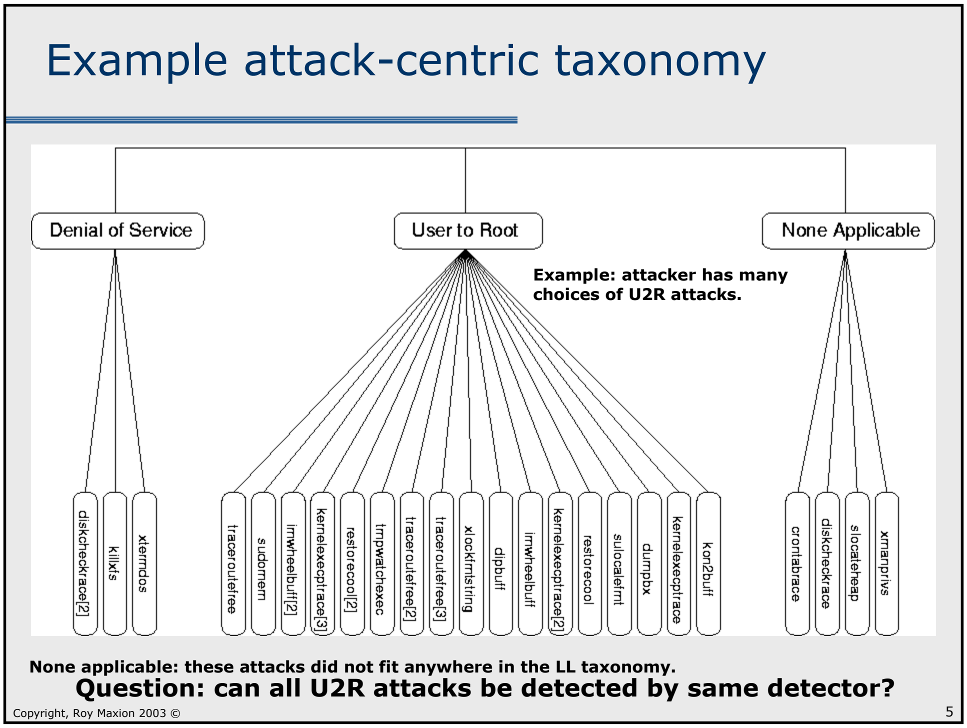
Observation

- Most of the known attack taxonomies are constructed from the attacker’s perspective; that is, they are attack-centric.
- For the attacker, if one attack fails, choose another attack from the same taxon; try again.
- Such taxonomies are great for the attacker … … but less attractive for the defender.



Goal

- Predict if an IDS will detect a given attack.
  - Need to know attack manifestations … but ...
  - Manifestations are not provided by any current taxonomies (closest is Kumar, but too abstract, and focused on signatures).
- Compare two taxonomies for equivalence.
  - Attack-centric and defense-centric.
  - Note that this is for anomaly detection … regarded as the best hope for detecting novel attacks, masquerade attacks, and other attacks detectable through profiling.



Example attack-centric taxonomy

Denial of Service
User to Root
None Applicable

diskcheckrace[2]
killxfs
xtermdos

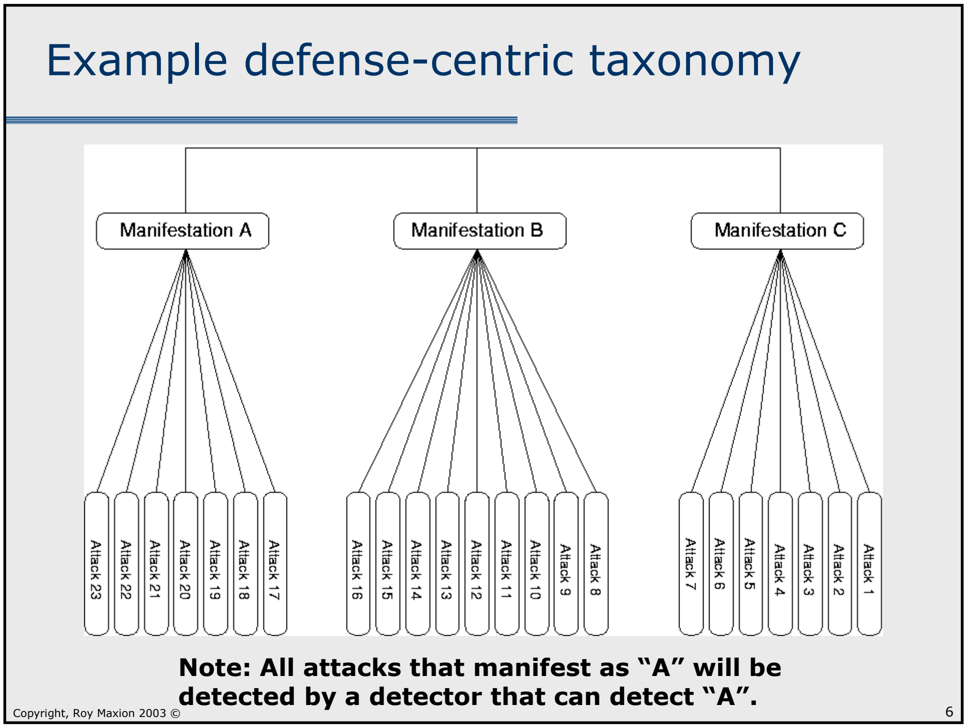
traceroutefree
sudomem
imwheelbuff[2]
kernelexecptrace[3]
restorecool[2]
tmpwatchexec
tracerouteftree[2]
tracerouteftree[3]
xlockfmtstring
dipbuff
imwheelbuff
kernelexecptrace[2]
restorecool
sulocalefmt
dumpbx
kernelexecptrace
kon2buff

crontabrace
diskcheckrace
slocateheap
xmanprivs

Example: attacker has many choices of U2R attacks.

None applicable: these attacks did not fit anywhere in the LL taxonomy.

Question: can all U2R attacks be detected by same detector?



Example defense-centric taxonomy

Manifestation A
Manifestation B
Manifestation C

Attack 23
Attack 22
Attack 21
Attack 20
Attack 19
Attack 18
Attack 17

Attack 16
Attack 15
Attack 14
Attack 13
Attack 12
Attack 11
Attack 10
Attack 9
Attack 8

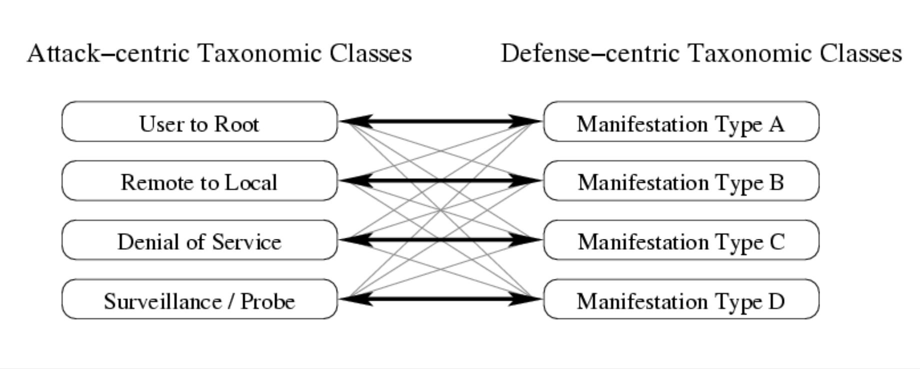
Attack 7
Attack 6
Attack 5
Attack 4
Attack 3
Attack 2
Attack 1

Note: All attacks that manifest as "A" will be detected by a detector that can detect "A".

What to do?

- Build a taxonomy that is defense-centric.
- Check that it obeys classic taxonomic rules.
- Assemble a collection of programs that could operate as attacks.
- Run these programs native to observe their normal behavior.
- Run them again, in attack mode, to observe their attack behavior (manifestations).
- Determine whether the manifestations mirror the classes of the attack-centric taxonomy, or not.

What to do … pictorially?

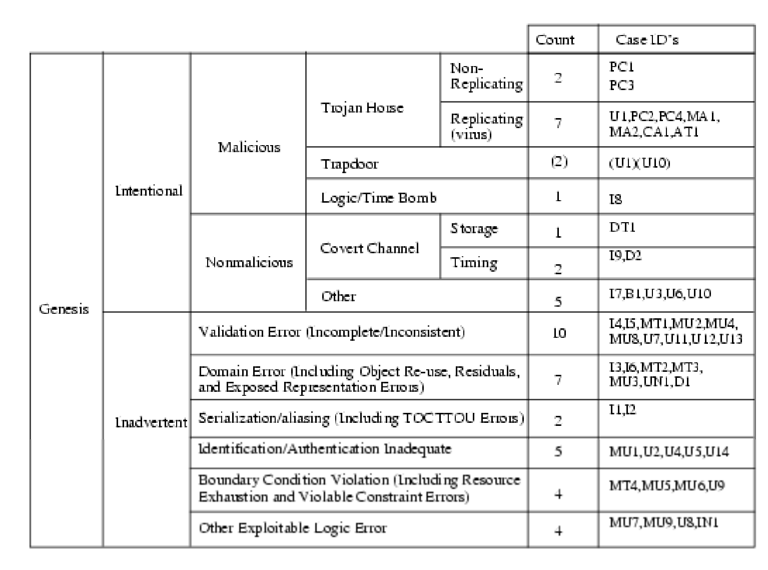


Do the attacks on the left map directly to the classes on the right?

Examples of extant taxonomies

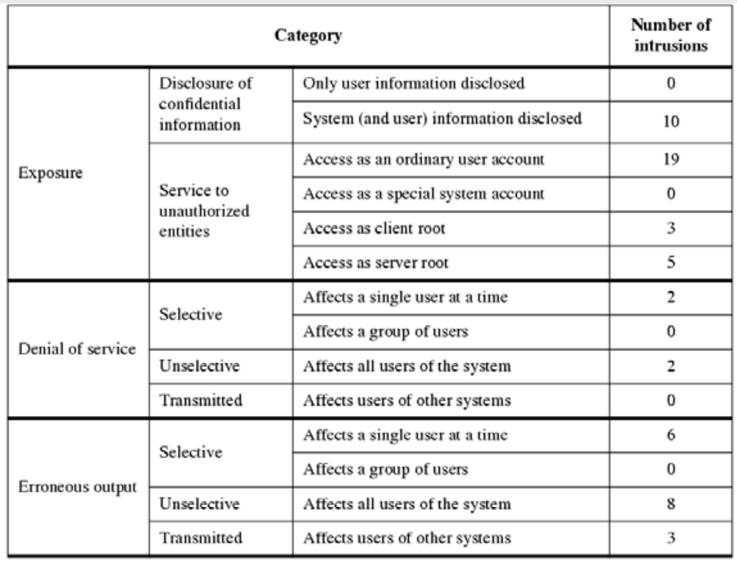
- Equivalence partitioning (Puketza)
  - Based on the likelihood of a particular IDS detecting an attack. (Not an actual taxonomy, but suggestive of one; could conceptually be used by a defender.)
- Flaw classifications (Landwehr)
  - Based on the kinds of programming flaws that facilitate attacks (e.g., buffer overflow)
- Attack classifications (Lindqvist & Jonsson)
  - Based on intended effect of attack (e.g., denial of service)
- Signature classifications (Kumar)
  - Based on the complexity of attack signatures
- Attack types (Lippmann et al. (LL))
  - Based on the intended effect of the attack (e.g., elevating user to root)

Landwehr et al. (flaw classifications)



					Count	Case ID's
Genesis	Intentional	Malicious	Trojan Horse	Non-Replicating	2	PC1 PC3
				Replicating (virus)	7	U1,PC2,PC4,MA1, MA2,CA1,AT1
			Trapdoor		(2)	(U1)(U10)
			Logic/Time Bomb		1	I8
		Nonmalicious	Covert Channel	Storage	1	DT1
				Timing	2	I9,D2
			Other		5	I7,B1,U3,U6,U10
	Inadvertent	Validation Error (Incomplete/Inconsistent)			10	I4,I5,MT1,MU2,MU4, MU8,U7,U11,U12,U13
		Domain Error (Including Object Re-use, Residuals, and Exposed Representation Errors)			7	I3,I6,MT2,MT3, MU3,UN1,D1
		Serialization/aliasing (Including TOCTTOU Errors)			2	I1,I2
		Identification/Authentication Inadequate			5	MU1,U2,U4,U5,U14
		Boundary Condition Violation (Including Resource Exhaustion and Violable Constraint Errors)			4	MT4,MU5,MU6,U9
		Other Exploitable Logic Error			4	MU7,MU9,U8,IN1

Lindqvist & Jonsson (attack classifications)



Category			Number of intrusions
Exposure	Disclosure of confidential information	Only user information disclosed	0
		System (and user) information disclosed	10
	Service to unauthorized entities	Access as an ordinary user account	19
		Access as a special system account	0
		Access as client root	3
		Access as server root	5
Denial of service	Selective	Affects a single user at a time	2
		Affects a group of users	0
	Unselective	Affects all users of the system	2
	Transmitted	Affects users of other systems	0
Erroneous output	Selective	Affects a single user at a time	6
		Affects a group of users	0
	Unselective	Affects all users of the system	8
	Transmitted	Affects users of other systems	3

Kumar (signature classifications)

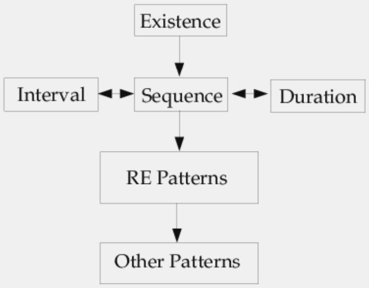
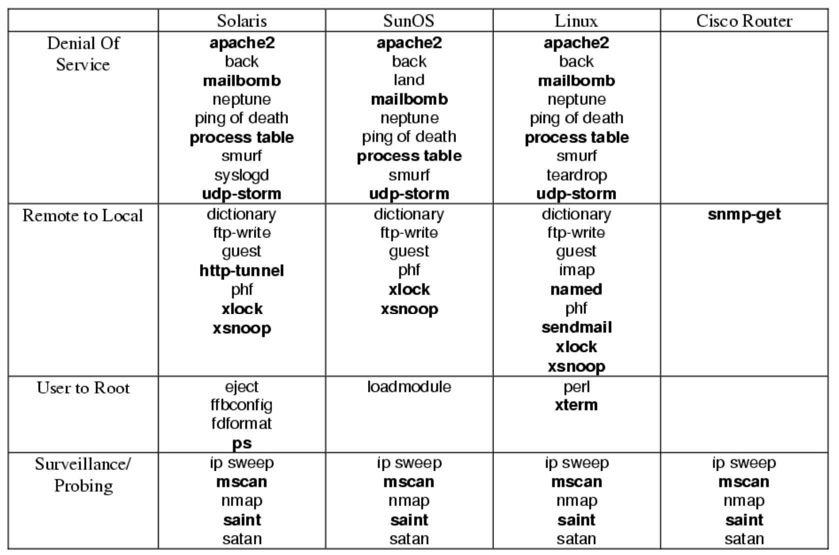


Figure 3.2 The Abstract Signature Classification Hierarchy

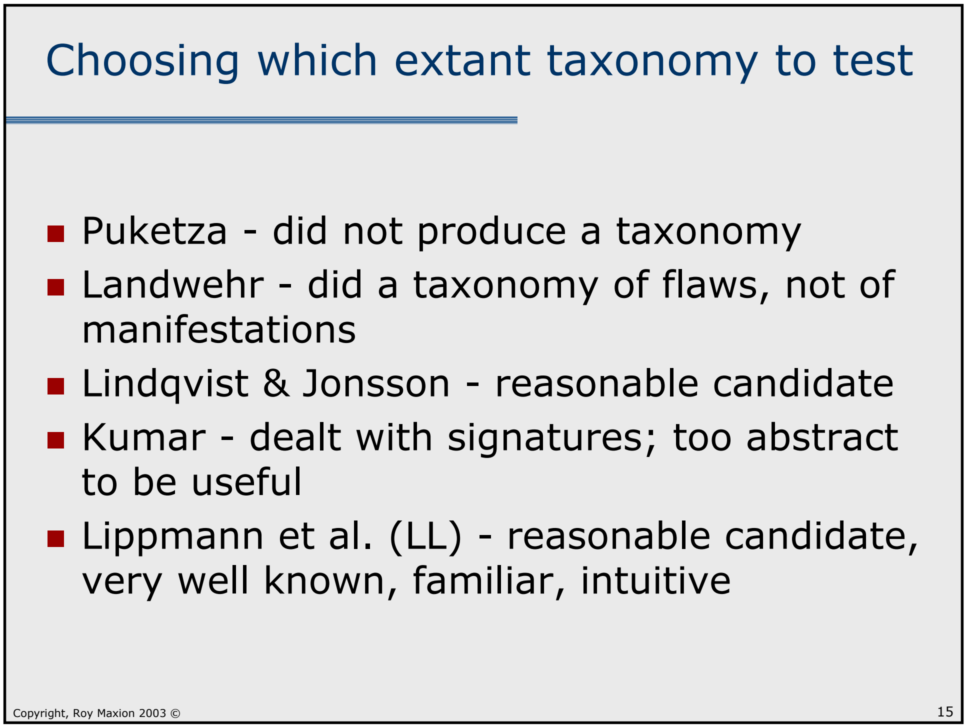
Lippmann et al. (LL) (attack types)



	Solaris	SunOS	Linux	Cisco Router
Denial Of Service	**apache2** back **mailbomb** neptune ping of death **process table** smurf syslogd **udp-storm**	**apache2** back land **mailbomb** neptune ping of death **process table** smurf **udp-storm**	**apache2** back **mailbomb** neptune ping of death **process table** smurf teardrop **udp-storm**	
Remote to Local	dictionary ftp-write guest **http-tunnel** phf **xlock** **xsnoop**	dictionary ftp-write guest phf **xlock** **xsnoop**	dictionary ftp-write guest imap **named** phf **sendmail** **xlock** **xsnoop**	**snmp-get**
User to Root	eject ffbconfig fdformat **ps**	loadmodule	perl **xterm**	
Surveillance/ Probing	ip sweep **mscan** nmap **saint** satan	ip sweep **mscan** nmap **saint** satan	ip sweep **mscan** nmap **saint** satan	ip sweep **mscan** nmap **saint** satan

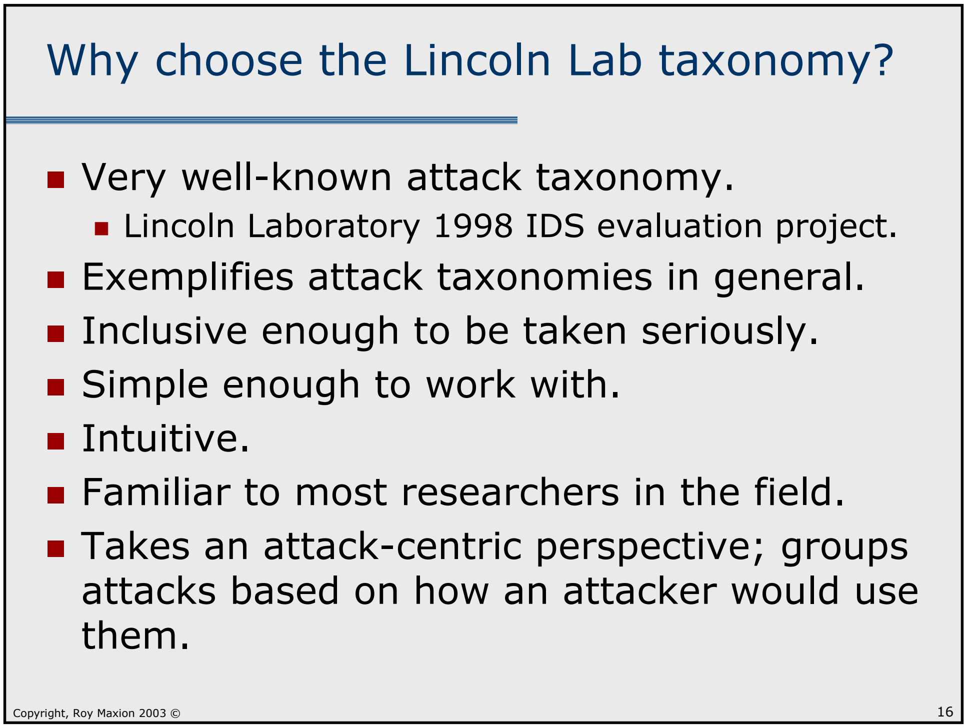
What's wrong with these taxonomies?

- Nothing is wrong with them.
- They just serve purposes different from the one we have in mind (except Kumar).
- Actually, we don't know if they serve our purpose or not, so we need to test the hypothesis that they do.
- But we only have time to test one of the taxonomies, so which one, and why?



Choosing which extant taxonomy to test

- Puketza - did not produce a taxonomy
- Landwehr - did a taxonomy of flaws, not of manifestations
- Lindqvist & Jonsson - reasonable candidate
- Kumar - dealt with signatures; too abstract to be useful
- Lippmann et al. (LL) - reasonable candidate, very well known, familiar, intuitive



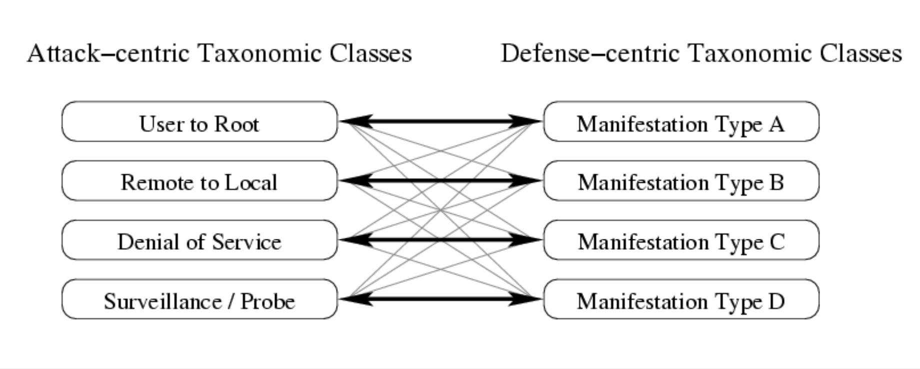
Why choose the Lincoln Lab taxonomy?

- Very well-known attack taxonomy.
  - Lincoln Laboratory 1998 IDS evaluation project.
- Exemplifies attack taxonomies in general.
- Inclusive enough to be taken seriously.
- Simple enough to work with.
- Intuitive.
- Familiar to most researchers in the field.
- Takes an attack-centric perspective; groups attacks based on how an attacker would use them.

What are the LL attack classes?

- User-to-root (U2R)
  - An attacker with the privileges of a regular user can use the attack to gain root privileges on the target machine.
- Remote-to-local (R2L)
  - An attacker with an Internet connection can use the attack to gain a local account on the target.
- Denial-of-service (DOS)
  - An attacker can use the attack to deny legitimate service to the target or a resource running on the target.
- Surveillance/probe (PRO)
  - An attacker can use the attack to gather reconnaissance information about the target, its users and resources.

Hypothesis: There is a 1-1 mapping.



Attack–centric Taxonomic Classes

Defense–centric Taxonomic Classes

User to Root

Remote to Local

Denial of Service

Surveillance / Probe

Manifestation Type A

Manifestation Type B

Manifestation Type C

Manifestation Type D

Do the attacks on the left map directly to the classes on the right?

Methodology

- Choose attack-centric taxonomy
- Develop attacker-defender testbed
- Develop attacks
- Gather normal traces
- Gather attack traces
- Extract attack manifestations
- Classify attacks according to manifestations and according to taxonomy under test
- Evaluate the mapping
- Acquire convergent evidence from IDS

Attacker/defender test bed

- Three (Intel-architecture-compatible) machines simulate the attacker’s and defender’s environments:
  - Attacker’s machine
  - Victim’s machine
  - Auxiliary machine (for support purposes, e.g., backup server for dump/restore U2R attack)
- RedHat Linux operating system
- Victim instrumentation: IMMSEC system call logger
  - Kernel patch
  - Logs all system calls

Example system-call data (truncated)

```
socket(PF_INET, SOCK_STREAM, IPPROTO_IP) = 4
bind(4, {sin_family=AF_INET, sin_port=htons(1023), sin_addr=inet_addr("0.0.0.0")}}, 16) = 0
connect(4, {sin_family=AF_INET, sin_port=htons(515), sin_addr=inet_addr("128.2.205.3")}}, 16) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 5), ...}) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40034000
ioctl(1, TCGETS, {B9600 opost isig icanon echo ...}) = 0
write(1, "yellow.srv.cs.cmu.edu...  ", 26) = 26
write(4, "\3slate\n", 7) = 7
read(4, "slate accepting requests since S"..., 8192) = 60
write(1, "slate accepting requests since S"..., 60) = 60
read(4, "slate-16951            root    "..., 8192) = 71
write(1, "slate-16951            root    "..., 71) = 71
read(4, "Rank   Owner      Job           "..., 8192) = 141
write(1, "Rank   Owner      Job           "..., 141) = 141
read(4, "", 8192) = 0
close(4) = 0
chdir("/usr/spool/lpd/slate") = 0
```

System programs used

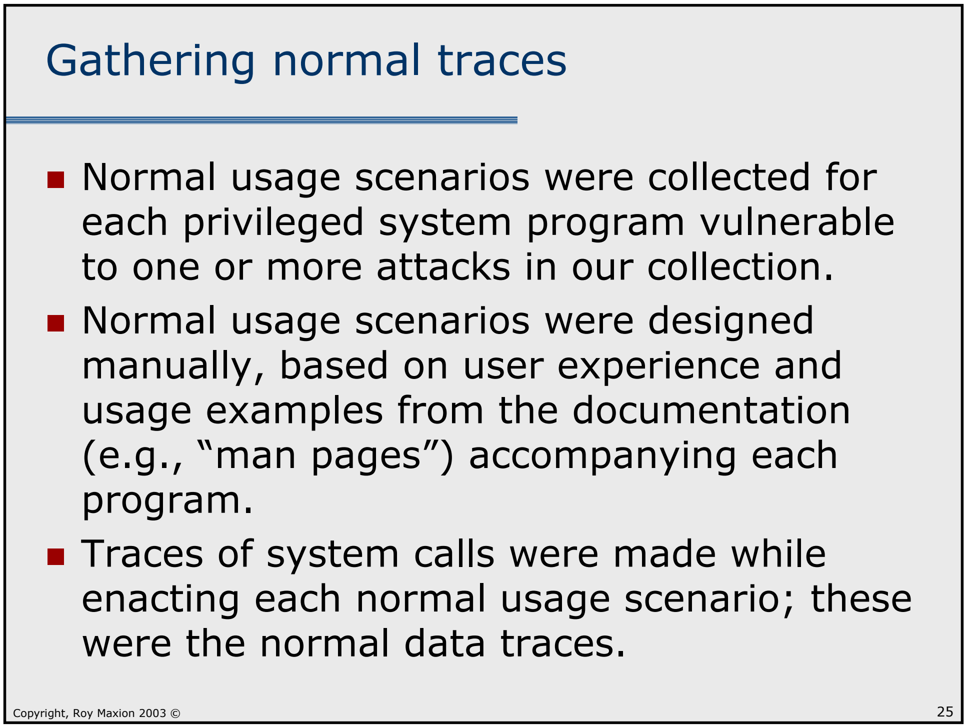
- These programs were run normally to get normal/training data.
- dip
- diskcheck
- dump
- imwheel
- kon2
- ntop
- passwd
- restore
- slocate
- slocate
- su
- sudo
- tmpwatch
- traceroute
- vim
- xfs
- xlock
- xman
- xterm

Developing the attacks

- Used vulnerability descriptions from public repositories (e.g., bugtraq)
- Attack criteria
  - Must involve exploitation of privileged system programs
  - Attack must actually work
- Attacks were modified from downloaded exploits, were developed from vulnerability descriptions, or were downloaded directly.
- Some attacks were designed to be difficult to detect, using hiding strategies (cloaking)

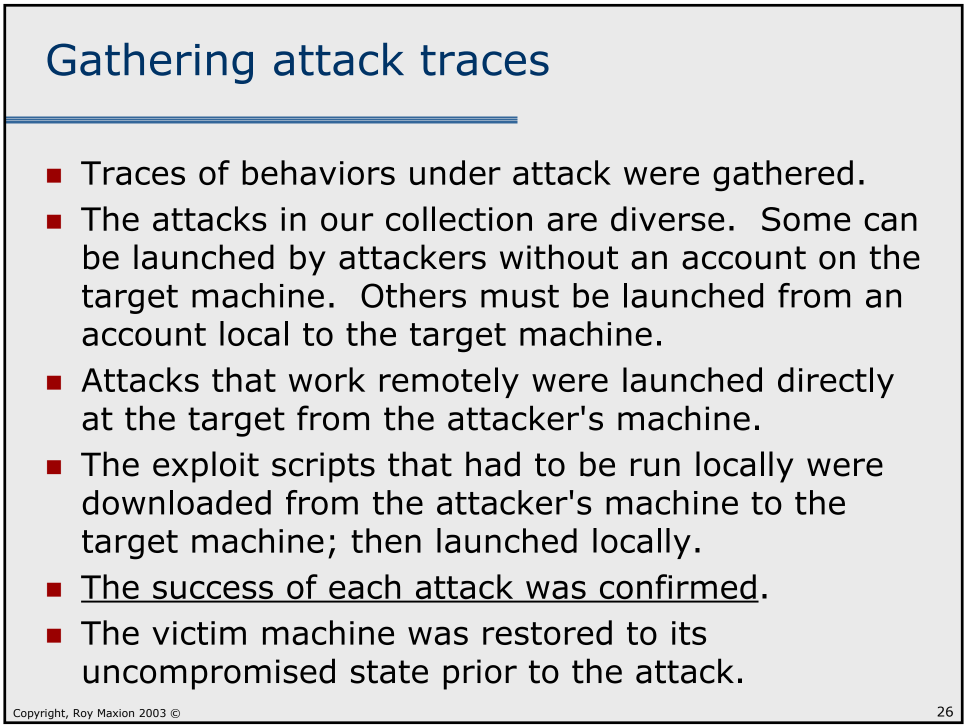
The attacks

- crontabrace
- dipbuff
- diskcheckrace
- diskcheckrace[2]
- dumpbx
- imwheelbuff
- imwheelbuff[2]
- kernelexecptrace
- kernelexecptrace[2]
- kernelexecptrace[3]
- killxfs
- kon2buff
- ntopspy
- restorecool
- restorecool[2]
- slocateheap
- sudomem
- sulocalefmt
- tmpwatchexec
- traceroutefree
- traceroutefree[2]
- traceroutefree[3]
- xlockfmtstring
- xmanprivs
- xtermdos



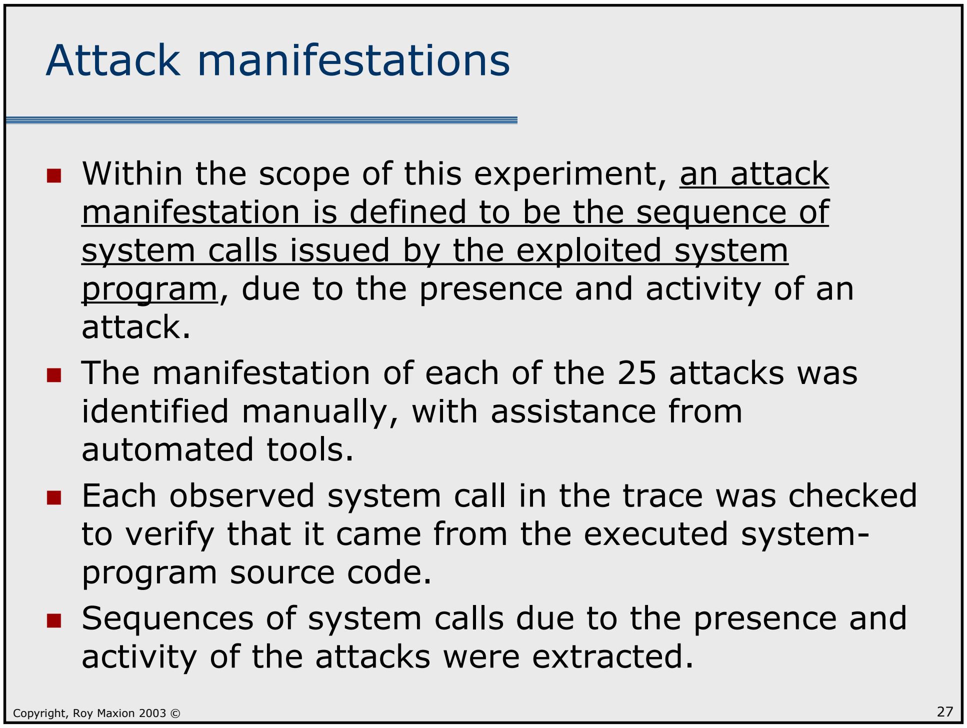
Gathering normal traces

- Normal usage scenarios were collected for each privileged system program vulnerable to one or more attacks in our collection.
- Normal usage scenarios were designed manually, based on user experience and usage examples from the documentation (e.g., “man pages”) accompanying each program.
- Traces of system calls were made while enacting each normal usage scenario; these were the normal data traces.



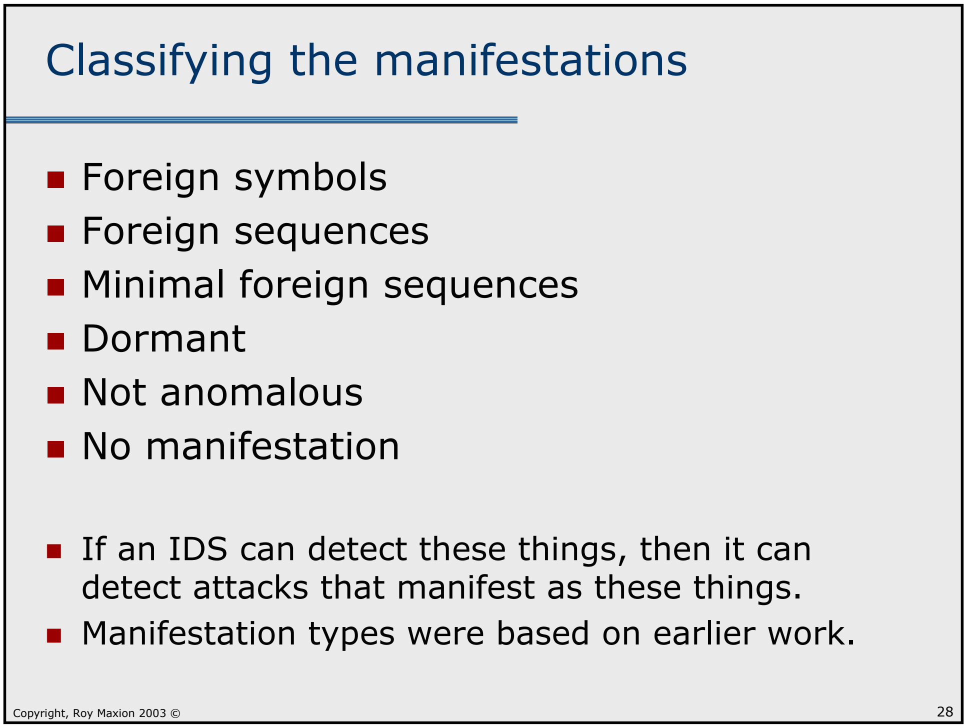
Gathering attack traces

- Traces of behaviors under attack were gathered.
- The attacks in our collection are diverse. Some can be launched by attackers without an account on the target machine. Others must be launched from an account local to the target machine.
- Attacks that work remotely were launched directly at the target from the attacker's machine.
- The exploit scripts that had to be run locally were downloaded from the attacker's machine to the target machine; then launched locally.
- The success of each attack was confirmed.
- The victim machine was restored to its uncompromised state prior to the attack.



Attack manifestations

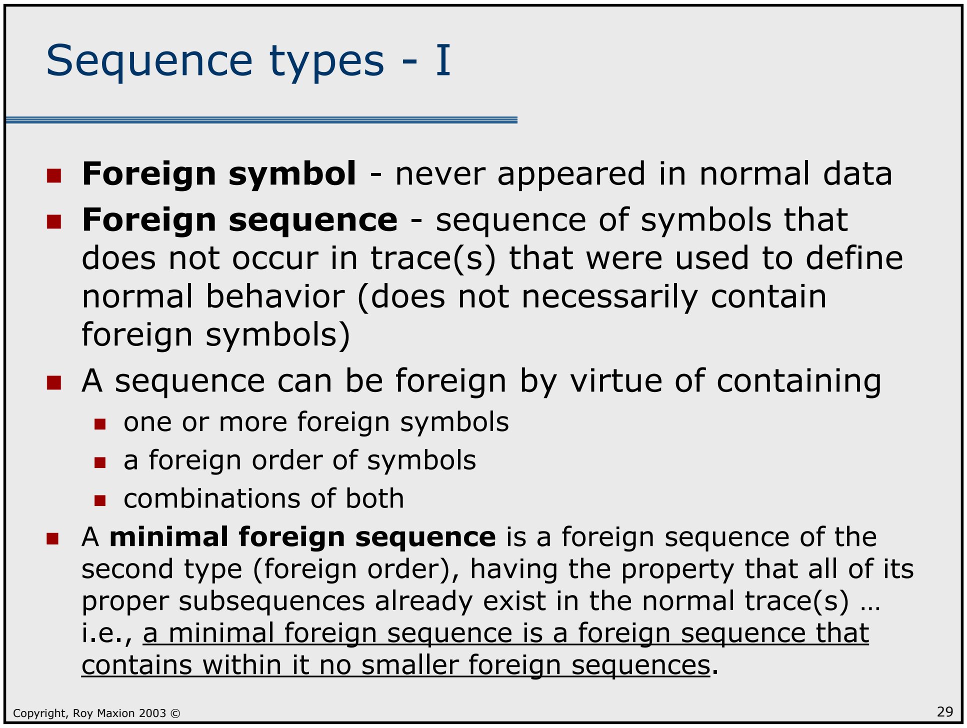
- Within the scope of this experiment, an attack manifestation is defined to be the sequence of system calls issued by the exploited system program, due to the presence and activity of an attack.
- The manifestation of each of the 25 attacks was identified manually, with assistance from automated tools.
- Each observed system call in the trace was checked to verify that it came from the executed system-program source code.
- Sequences of system calls due to the presence and activity of the attacks were extracted.



Classifying the manifestations

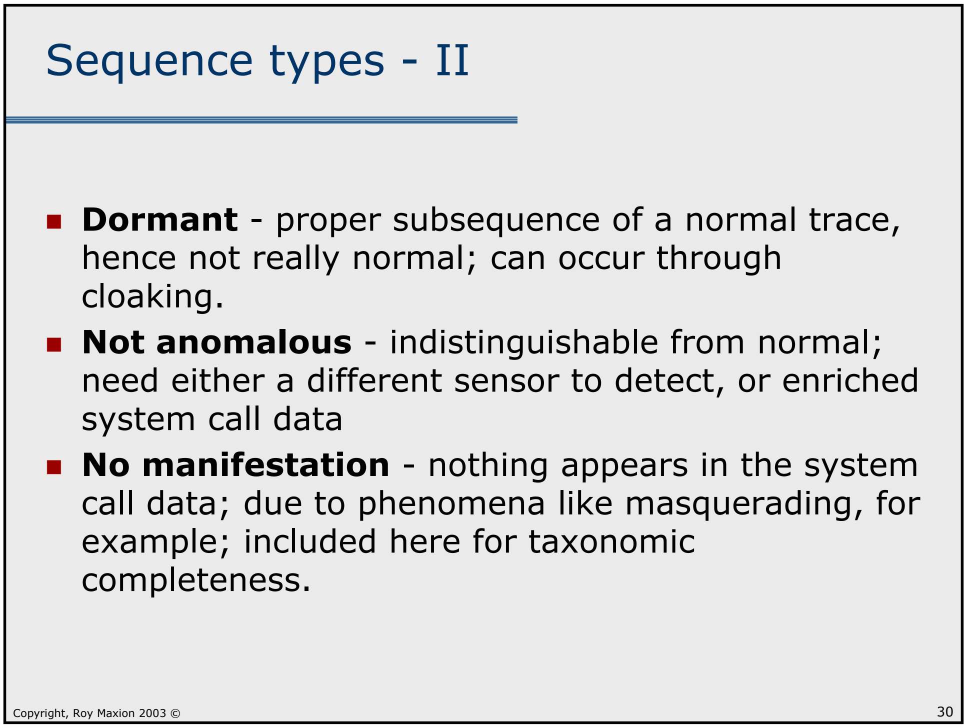
- Foreign symbols
- Foreign sequences
- Minimal foreign sequences
- Dormant
- Not anomalous
- No manifestation

- If an IDS can detect these things, then it can detect attacks that manifest as these things.
- Manifestation types were based on earlier work.



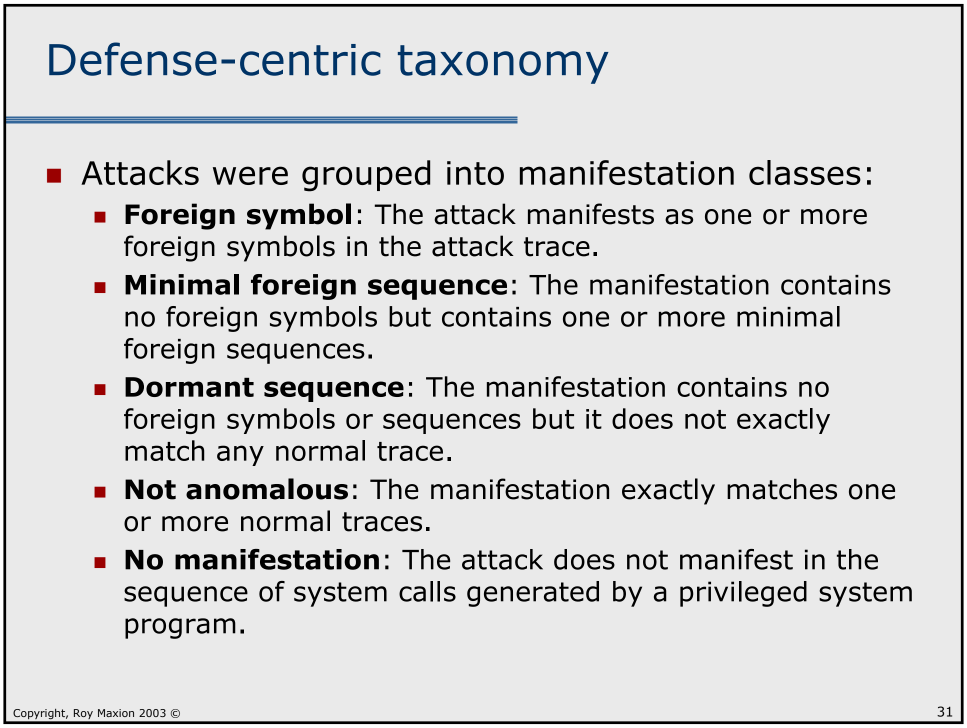
Sequence types - I

- **Foreign symbol** - never appeared in normal data
- **Foreign sequence** - sequence of symbols that does not occur in trace(s) that were used to define normal behavior (does not necessarily contain foreign symbols)
- A sequence can be foreign by virtue of containing
  - one or more foreign symbols
  - a foreign order of symbols
  - combinations of both
- A **minimal foreign sequence** is a foreign sequence of the second type (foreign order), having the property that all of its proper subsequences already exist in the normal trace(s) … i.e., a minimal foreign sequence is a foreign sequence that contains within it no smaller foreign sequences.



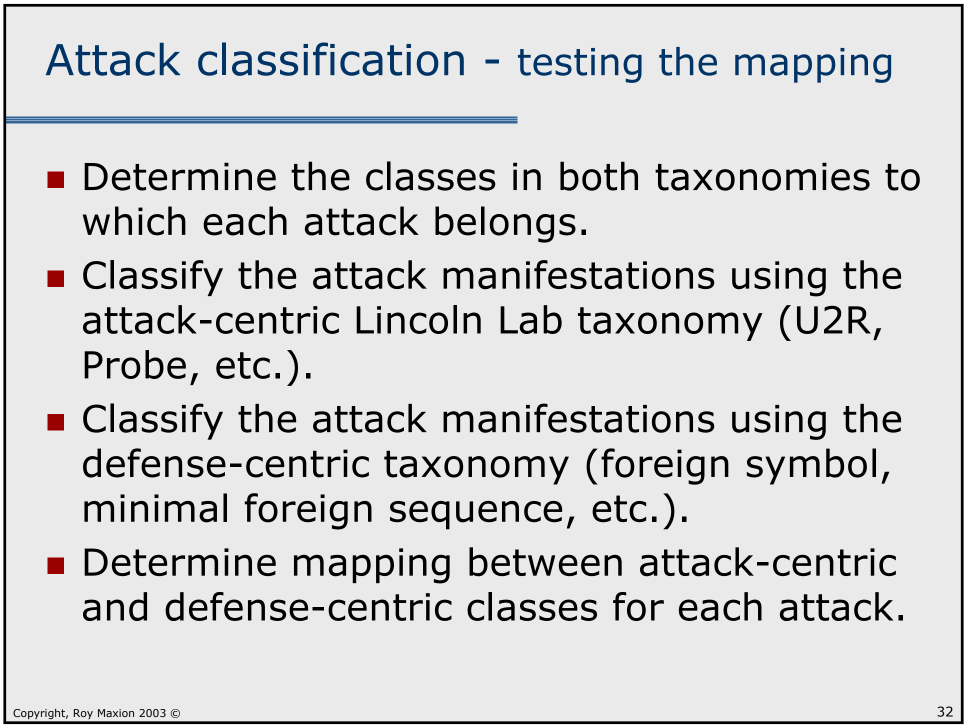
Sequence types - II

- **Dormant** - proper subsequence of a normal trace, hence not really normal; can occur through cloaking.
- **Not anomalous** - indistinguishable from normal; need either a different sensor to detect, or enriched system call data
- **No manifestation** - nothing appears in the system call data; due to phenomena like masquerading, for example; included here for taxonomic completeness.



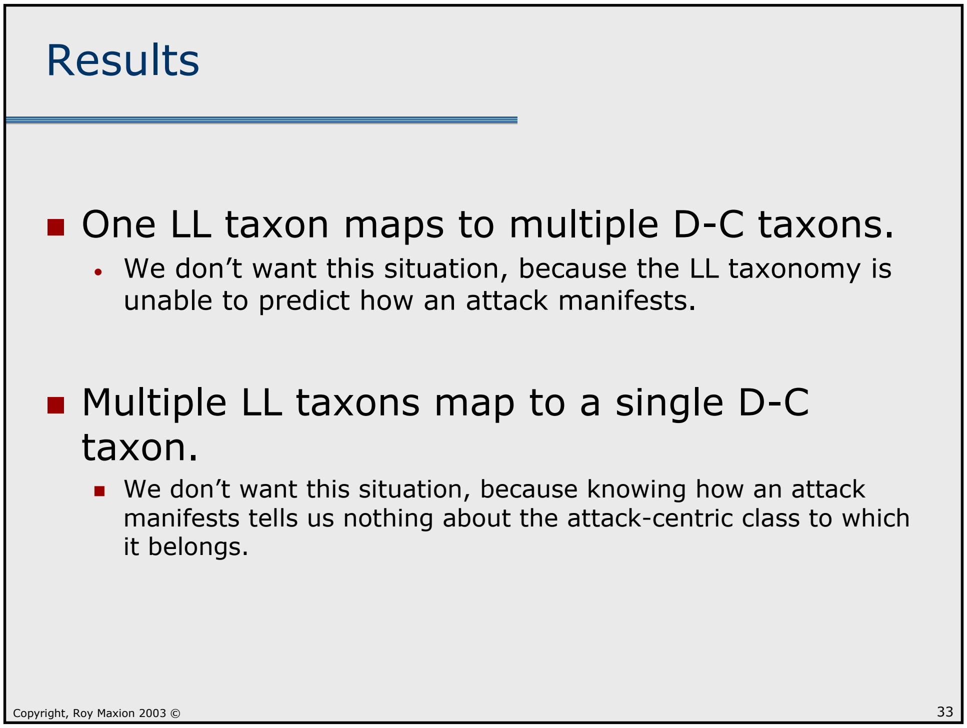
Defense-centric taxonomy

- Attacks were grouped into manifestation classes:
  - **Foreign symbol**: The attack manifests as one or more foreign symbols in the attack trace.
  - **Minimal foreign sequence**: The manifestation contains no foreign symbols but contains one or more minimal foreign sequences.
  - **Dormant sequence**: The manifestation contains no foreign symbols or sequences but it does not exactly match any normal trace.
  - **Not anomalous**: The manifestation exactly matches one or more normal traces.
  - **No manifestation**: The attack does not manifest in the sequence of system calls generated by a privileged system program.



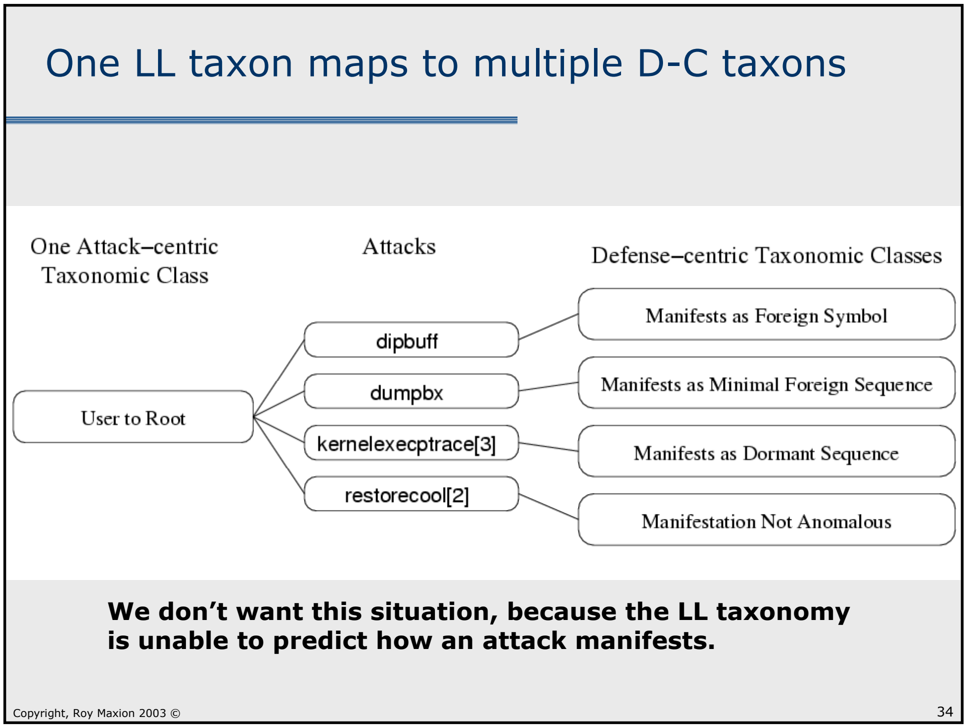
Attack classification - testing the mapping

- Determine the classes in both taxonomies to which each attack belongs.
- Classify the attack manifestations using the attack-centric Lincoln Lab taxonomy (U2R, Probe, etc.).
- Classify the attack manifestations using the defense-centric taxonomy (foreign symbol, minimal foreign sequence, etc.).
- Determine mapping between attack-centric and defense-centric classes for each attack.



Results

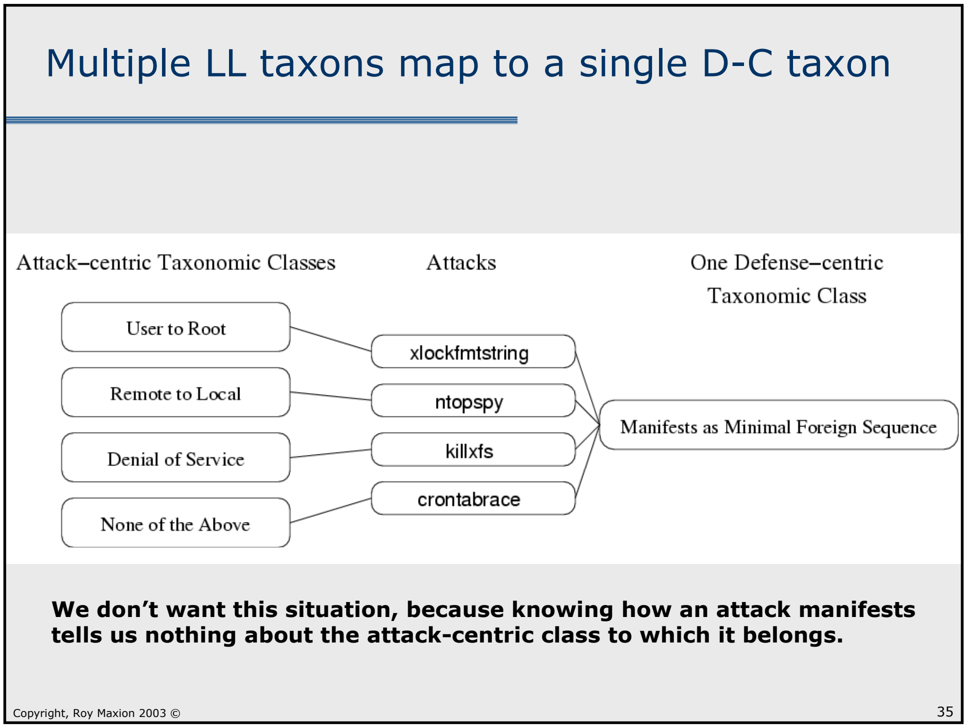
- One LL taxon maps to multiple D-C taxons.
  - We don’t want this situation, because the LL taxonomy is unable to predict how an attack manifests.
- Multiple LL taxons map to a single D-C taxon.
  - We don’t want this situation, because knowing how an attack manifests tells us nothing about the attack-centric class to which it belongs.



One LL taxon maps to multiple D-C taxons

One Attack–centric Taxonomic Class
Attacks
Defense–centric Taxonomic Classes
User to Root
dipbuff
dumpbx
kernelexecptrace[3]
restorecool[2]
Manifests as Foreign Symbol
Manifests as Minimal Foreign Sequence
Manifests as Dormant Sequence
Manifestation Not Anomalous

We don’t want this situation, because the LL taxonomy is unable to predict how an attack manifests.



Multiple LL taxons map to a single D-C taxon

Attack–centric Taxonomic Classes

Attacks

One Defense–centric Taxonomic Class

User to Root

Remote to Local

Denial of Service

None of the Above

xlockfmtstring

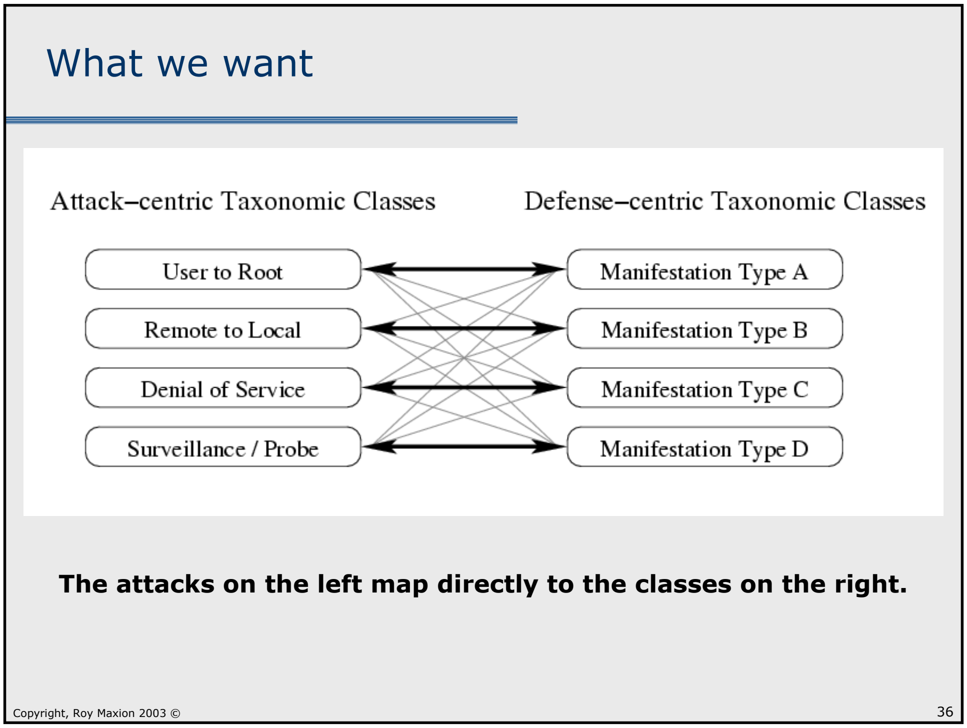
ntopspy

killxfs

crontabrace

Manifests as Minimal Foreign Sequence

We don’t want this situation, because knowing how an attack manifests tells us nothing about the attack-centric class to which it belongs.



What we want

Attack–centric Taxonomic Classes

Defense–centric Taxonomic Classes

User to Root

Remote to Local

Denial of Service

Surveillance / Probe

Manifestation Type A

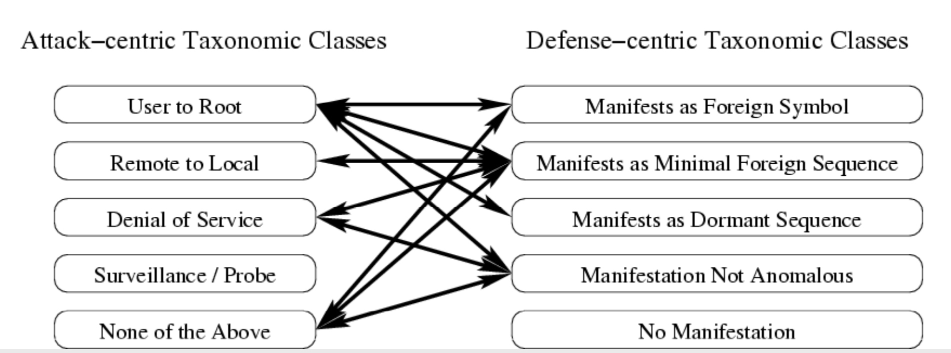
Manifestation Type B

Manifestation Type C

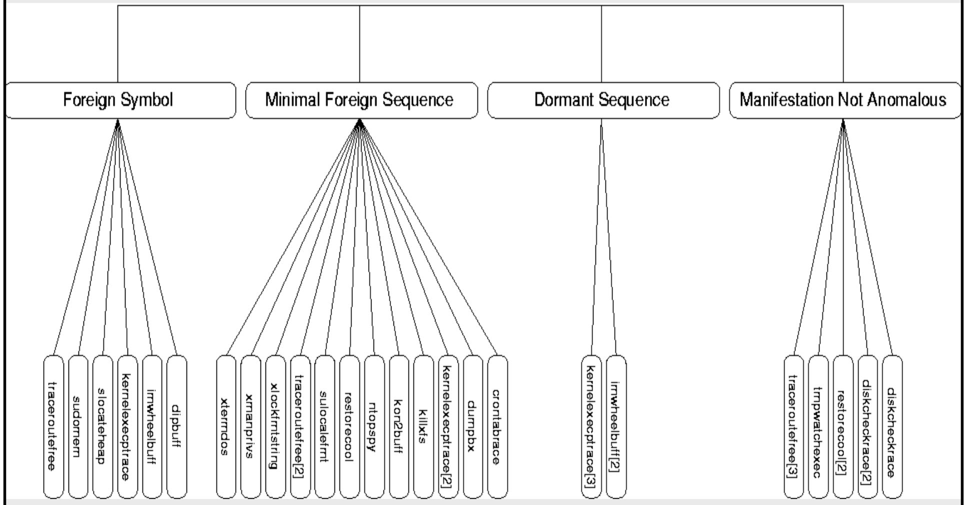
Manifestation Type D

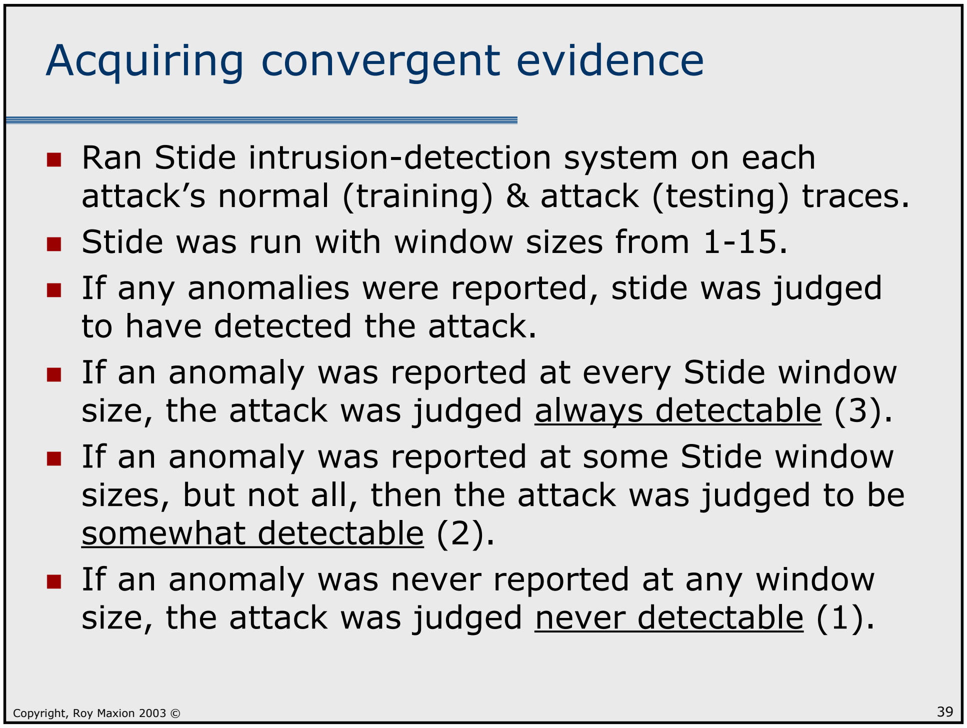
The attacks on the left map directly to the classes on the right.

What we got … as empirical correspondences


Attack–centric Taxonomic Classes
Defense–centric Taxonomic Classes
User to Root
Remote to Local
Denial of Service
Surveillance / Probe
None of the Above
Manifests as Foreign Symbol
Manifests as Minimal Foreign Sequence
Manifests as Dormant Sequence
Manifestation Not Anomalous
No Manifestation

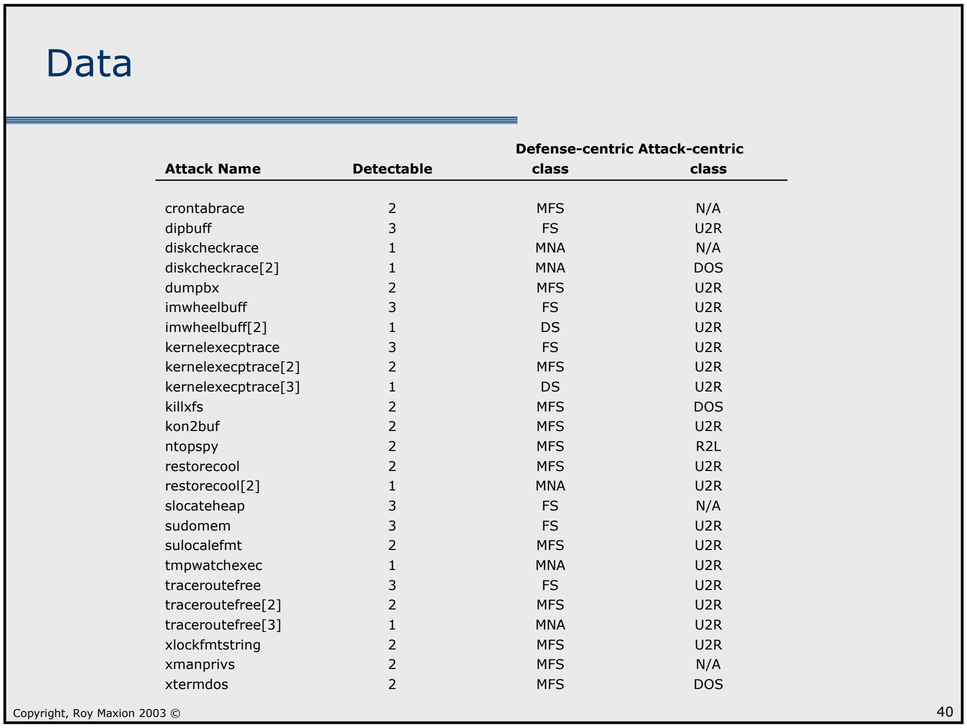
Full defense-centric taxonomy





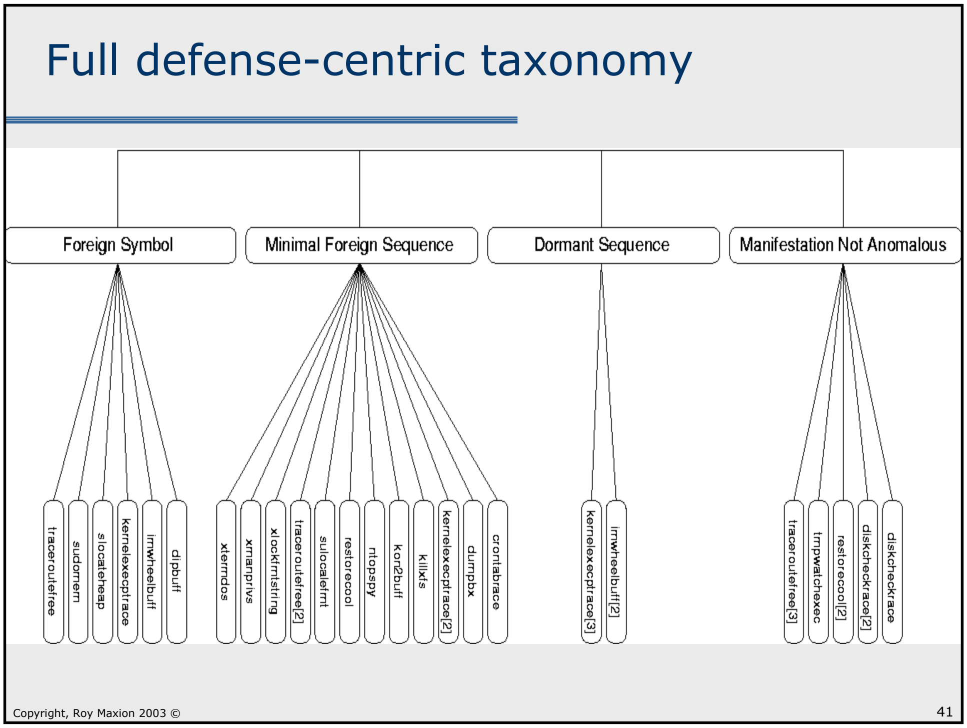
Acquiring convergent evidence

- Ran Stide intrusion-detection system on each attack’s normal (training) & attack (testing) traces.
- Stide was run with window sizes from 1-15.
- If any anomalies were reported, stide was judged to have detected the attack.
- If an anomaly was reported at every Stide window size, the attack was judged always detectable (3).
- If an anomaly was reported at some Stide window sizes, but not all, then the attack was judged to be somewhat detectable (2).
- If an anomaly was never reported at any window size, the attack was judged never detectable (1).



Data

Attack Name	Detectable	Defense-centric class	Attack-centric class
crontabrace	2	MFS	N/A
dipbuff	3	FS	U2R
diskcheckrace	1	MNA	N/A
diskcheckrace[2]	1	MNA	DOS
dumpbx	2	MFS	U2R
imwheelbuff	3	FS	U2R
imwheelbuff[2]	1	DS	U2R
kernelexecptrace	3	FS	U2R
kernelexecptrace[2]	2	MFS	U2R
kernelexecptrace[3]	1	DS	U2R
killxfs	2	MFS	DOS
kon2buf	2	MFS	U2R
ntopspy	2	MFS	R2L
restorecool	2	MFS	U2R
restorecool[2]	1	MNA	U2R
slocateheap	3	FS	N/A
sudomem	3	FS	U2R
sulocalefmt	2	MFS	U2R
tmpwatchexec	1	MNA	U2R
traceroutefree	3	FS	U2R
traceroutefree[2]	2	MFS	U2R
traceroutefree[3]	1	MNA	U2R
xlockfmtstring	2	MFS	U2R
xmanprivs	2	MFS	N/A
xtermdos	2	MFS	DOS



Full defense-centric taxonomy

Foreign Symbol

- traceroutefree
- sudomem
- slocateheap
- kernelexecptrace
- imwheelbuff
- dipbuff

Minimal Foreign Sequence

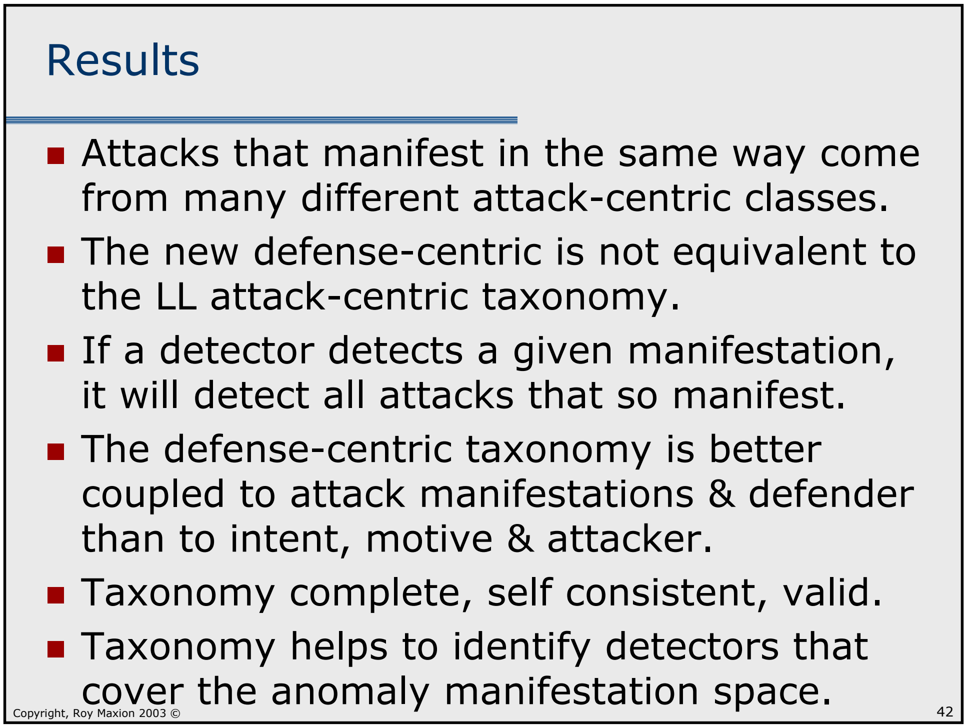
- xtermdos
- xmanprivs
- xlockfmtstring
- traceroutefree[2]
- sulocalefmt
- restorecool
- ntopspy
- kon2buff
- killxfs
- kernelexecptrace[2]
- dumpbx
- crontabrace

Dormant Sequence

- kernelexecptrace[3]
- imwheelbuff[2]

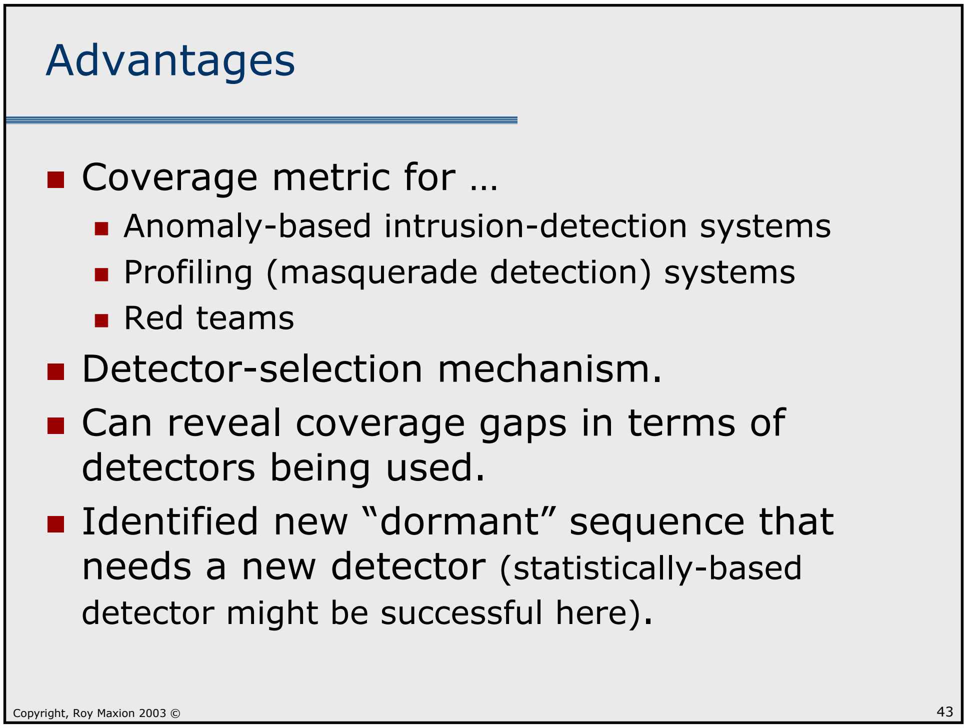
Manifestation Not Anomalous

- traceroutefree[3]
- tmpwatchexec
- restorecool[2]
- diskcheckrace[2]
- diskcheckrace



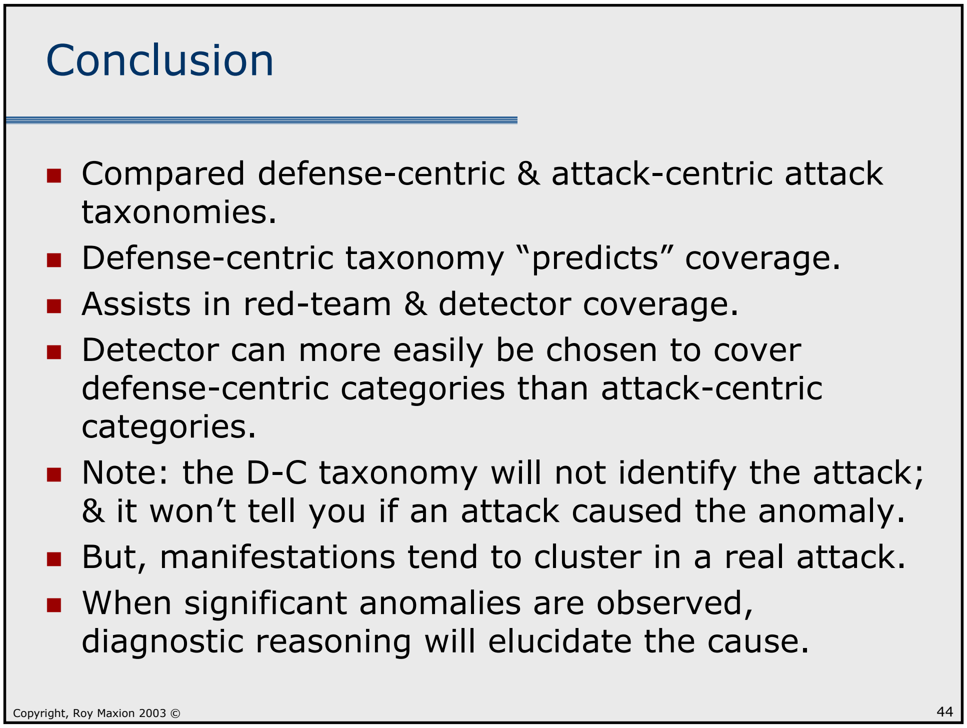
Results

- Attacks that manifest in the same way come from many different attack-centric classes.
- The new defense-centric is not equivalent to the LL attack-centric taxonomy.
- If a detector detects a given manifestation, it will detect all attacks that so manifest.
- The defense-centric taxonomy is better coupled to attack manifestations & defender than to intent, motive & attacker.
- Taxonomy complete, self consistent, valid.
- Taxonomy helps to identify detectors that cover the anomaly manifestation space.



Advantages

- Coverage metric for …
  - Anomaly-based intrusion-detection systems
  - Profiling (masquerade detection) systems
  - Red teams
- Detector-selection mechanism.
- Can reveal coverage gaps in terms of detectors being used.
- Identified new “dormant” sequence that needs a new detector (statistically-based detector might be successful here).



Conclusion

- Compared defense-centric & attack-centric attack taxonomies.
- Defense-centric taxonomy “predicts” coverage.
- Assists in red-team & detector coverage.
- Detector can more easily be chosen to cover defense-centric categories than attack-centric categories.
- Note: the D-C taxonomy will not identify the attack; & it won’t tell you if an attack caused the anomaly.
- But, manifestations tend to cluster in a real attack.
- When significant anomalies are observed, diagnostic reasoning will elucidate the cause.

- END - END - END - END - END - END -