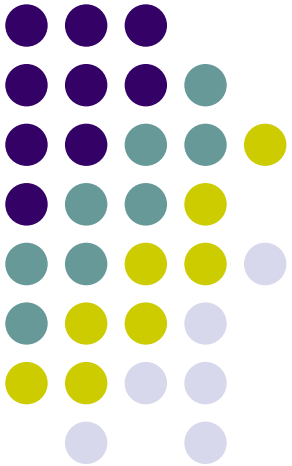
TOCTTOU Vulnerabilities in UNIX-Style File Systems: An Anatomical Study

Jinpeng Wei and Calton Pu

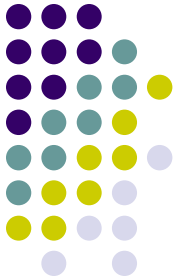
Georgia Institute of Technology



4th USENIX Conference on File and Storage Technologies

December 15, 2005. San Francisco, CA

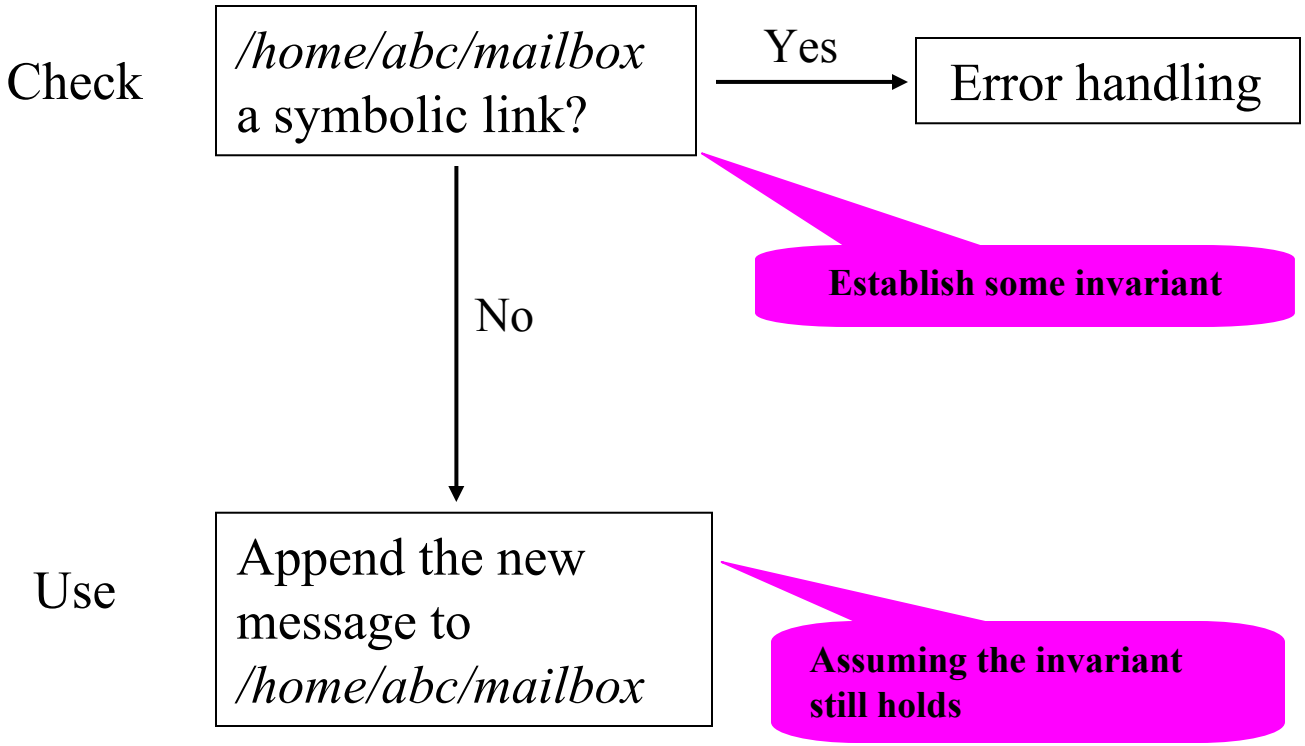
Definition and Scope



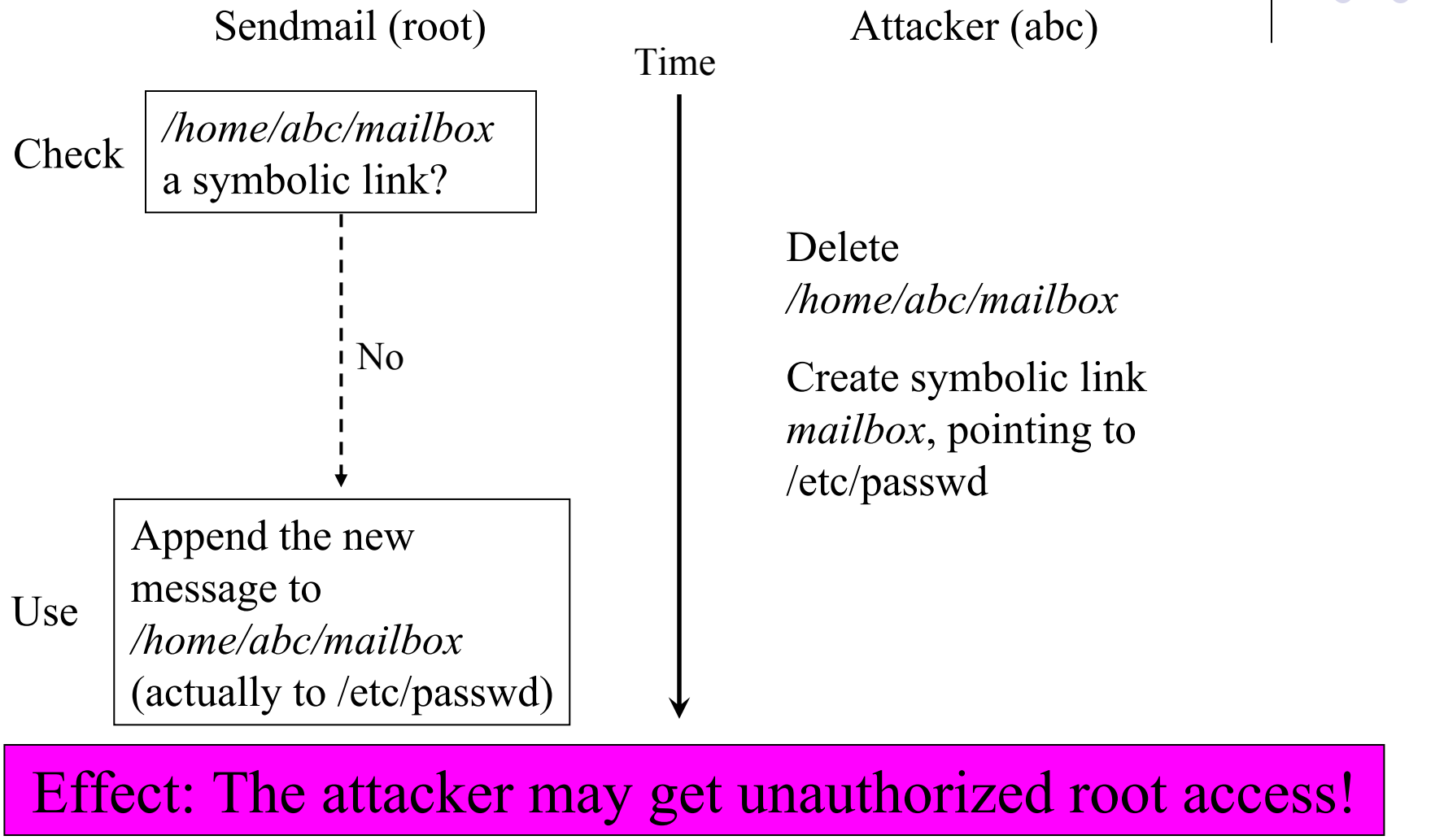
- TOCTTOU – Time of Check To Time of Use, a kind of race condition in Unix-style file systems
- Check – Establish some precondition (invariant) about a file
- Use – Operate on the file assuming that the invariant is still valid

Sendmail Example

- Run as root
- Operate on files owned by normal users

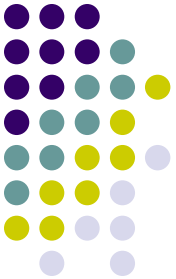


Sendmail Vulnerability: An Example



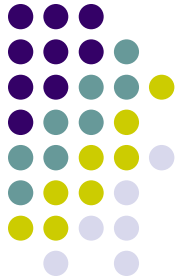
Effect: The attacker may get unauthorized root access!

Outline of the Presentation



- Motivation of the research
- CUU Model of TOCTTOU vulnerabilities
- Detection of TOCTTOU vulnerabilities
- Probability and event analysis of exploiting TOCTTOU vulnerabilities
- Related work and conclusion

CUU Model (1)

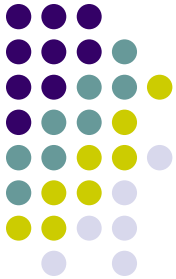


- CU-call: a system call that establishes some preconditions about a file, either explicitly or implicitly.
- Example CU-calls:

access, stat, open, creat, mkdir, rmdir

```
int stat(const char *file_name, struct stat *buf);
```

CUU Model (2)



- Use-call: a system call that operates on a file.
- Example Use-calls:

  open, truncate, mkdir, rmdir, chdir, execve, chmod, chown

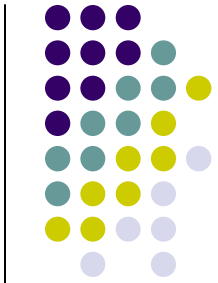
- A TOCTTOU pair is a combination of a CU-call and a Use-call. e.g., <stat, open> in Sendmail.

CUU Model (3)

Use	Explicit check	Implicit check
Create a regular file	CheckSet × FileCreationSet	FileRemovalSet × FileCreationSet
Create a directory	CheckSet × DirCreationSet	DirRemovalSet × DirCreationSet
Create a link	CheckSet × LinkCreationSet	LinkRemovalSet × LinkCreationSet
Read/Write/Execute or Change the attribute of a regular file	CheckSet × FileNormalUseSet	(FileCreationSet × FileNormalUseSet)∪ (LinkCreationSet × FileNormalUseSet)∪ (FileNormalUseSet × FileNormalUseSet)
Access or change the attribute of a directory	CheckSet × DirNormalUseSet	(DirCreationSet × DirNormalUseSet)∪ (LinkCreationSet × DirNormalUseSet)∪ (DirNormalUseSet × DirNormalUseSet)

CreationSet = FileCreationSet ∪ LinkCreationSet ∪ DirCreationSet

Concrete Sets For Linux



FileCreationSet = {creat, open, mknod, rename}

LinkCreationSet = {link, symlink, rename}

DirCreationSet = {mkdir, rename}

FileRemovalSet = {unlink, rename}

LinkRemovalSet = {unlink, rename}

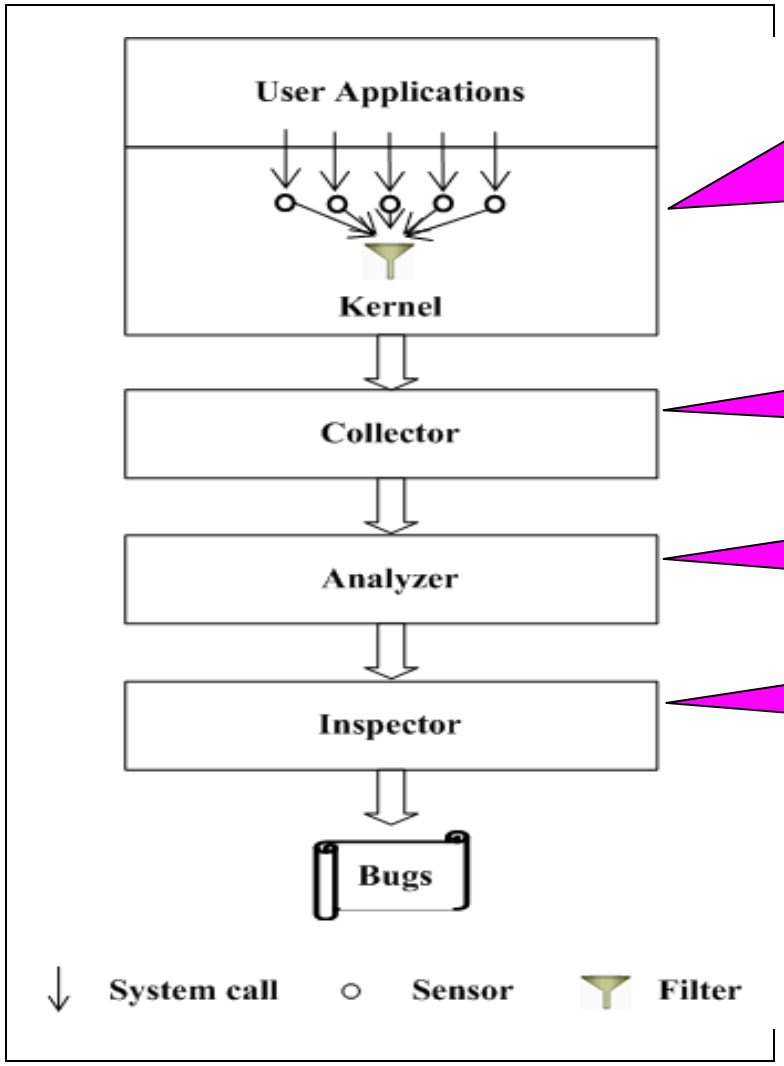
DirRemovalSet = {rmdir, rename}

FileNormalUseSet = {chmod, chown, truncate, utime, open, execve}

DirNormalUseSet = {chmod, chown, utime, mount, chdir, chroot, pivot_root}

CheckSet = {stat, access}

Detection Framework



The *Sensors* record file system call traces. The *filter* suppresses information about safe

Gathers kernel traces into log files

Identifies TOCTTOU pairs

Detects the TOCTTOU vulnerabilities

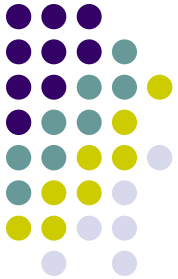
TOCTTOU Vulnerabilities in Red Hat Linux 9

Tested:

~130 utilities from /bin, /sbin and /usr/bin

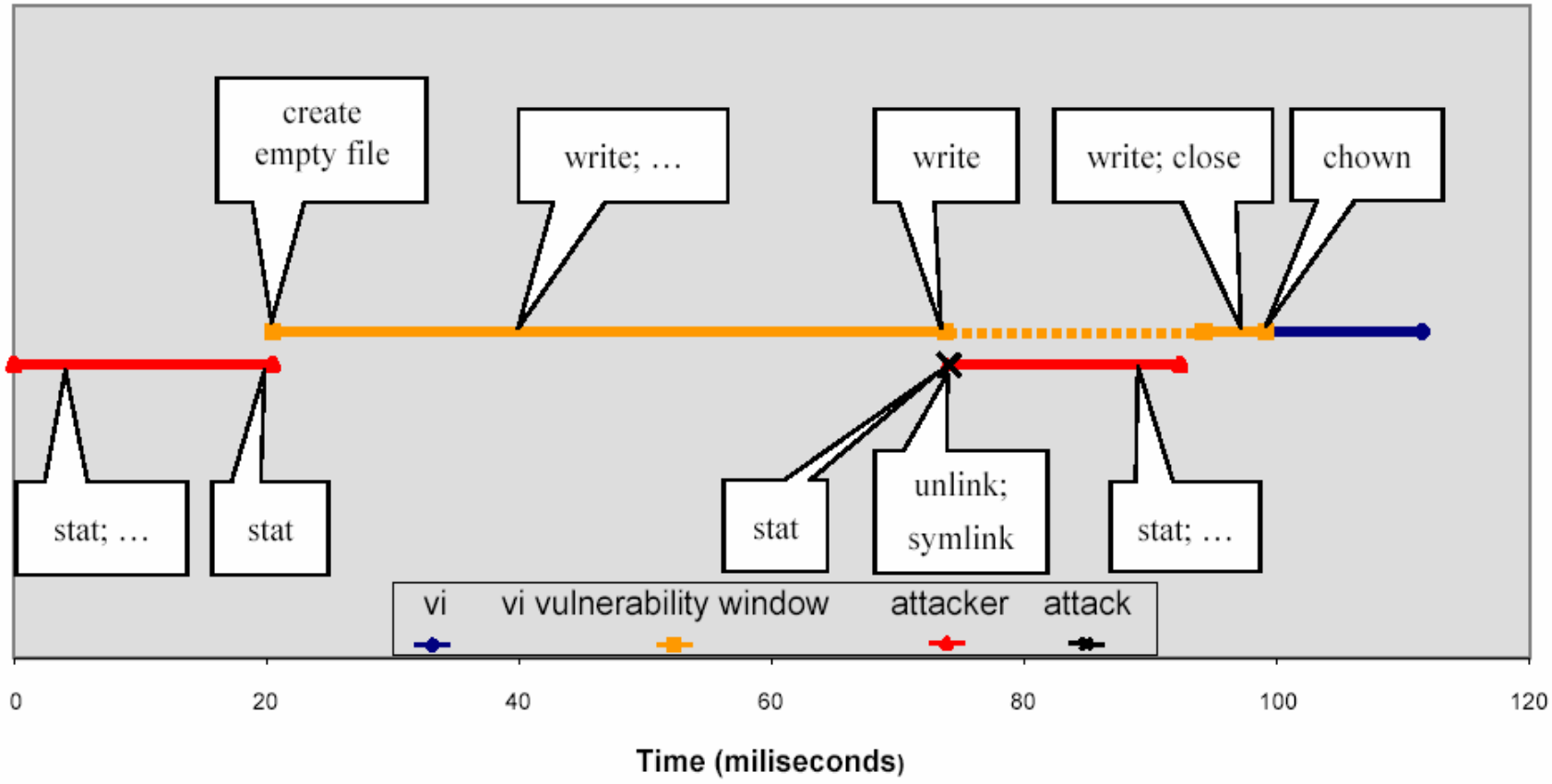
Application	TOCTTOU errors	Possible exploit
vi	**<open, chown>**	Changing the owner of /etc/passwd to an ordinary user
rpm	**<open, open>**	Running arbitrary command
emacs	**<open,chmod>**	Making /etc/shadow readable by an ordinary user
gedit	**<rename, chown>**	Changing the owner of /etc/passwd to an ordinary user
esd	**<mkdir, chmod>**	Gaining full access to another user's home directory

Vi 6.1 Vulnerability

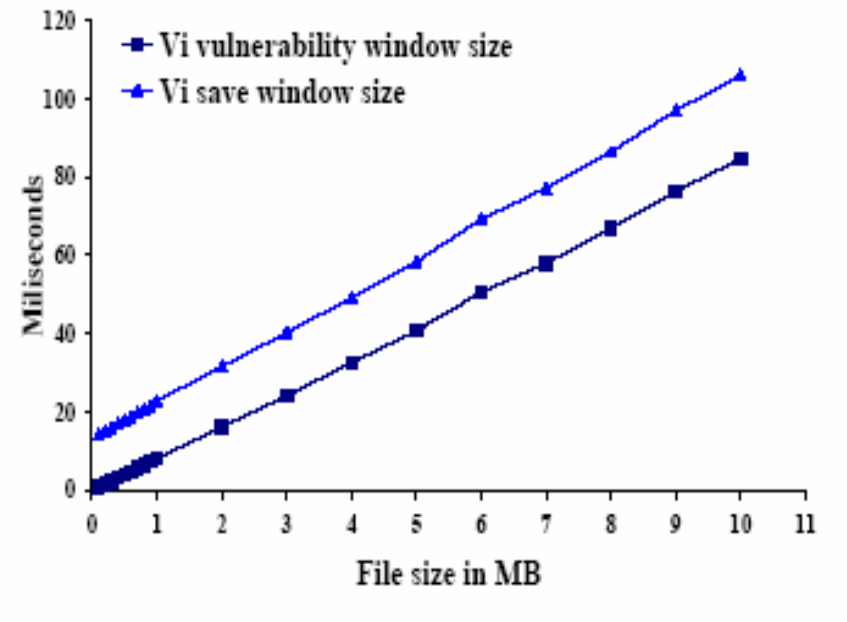


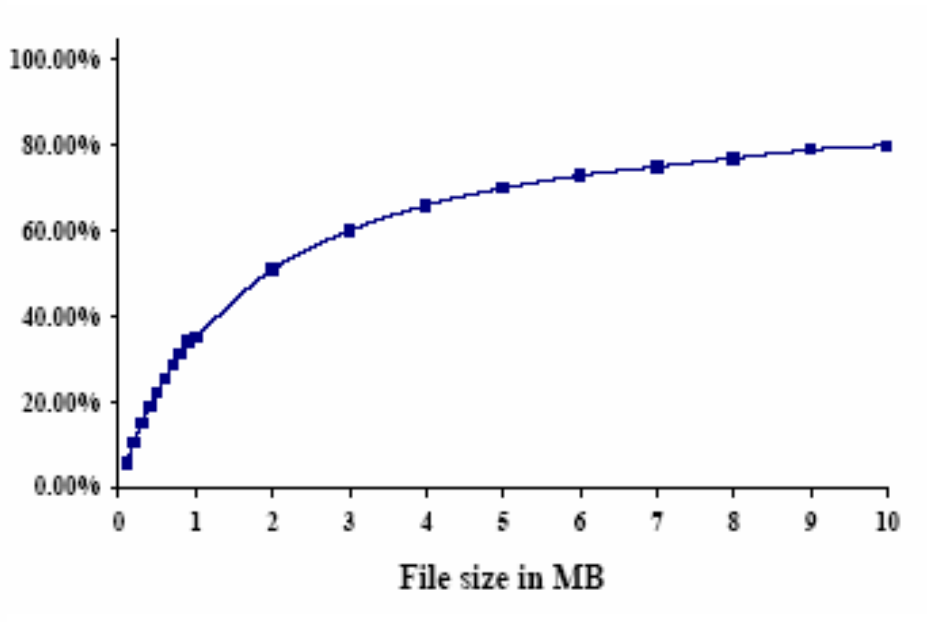
- The vulnerability happens when
  - vi is run by root
  - vi is editing a file owned by a normal user (can be an attacker)
  - vi saves the file being edited
- TOCTTOU pair: <**open, chown**>
  - **open** creates a new file for writing
  - **chown** changes the owner of the new file to the normal user.

Event Analysis of Vi Exploit



Baseline Vulnerability of Vi

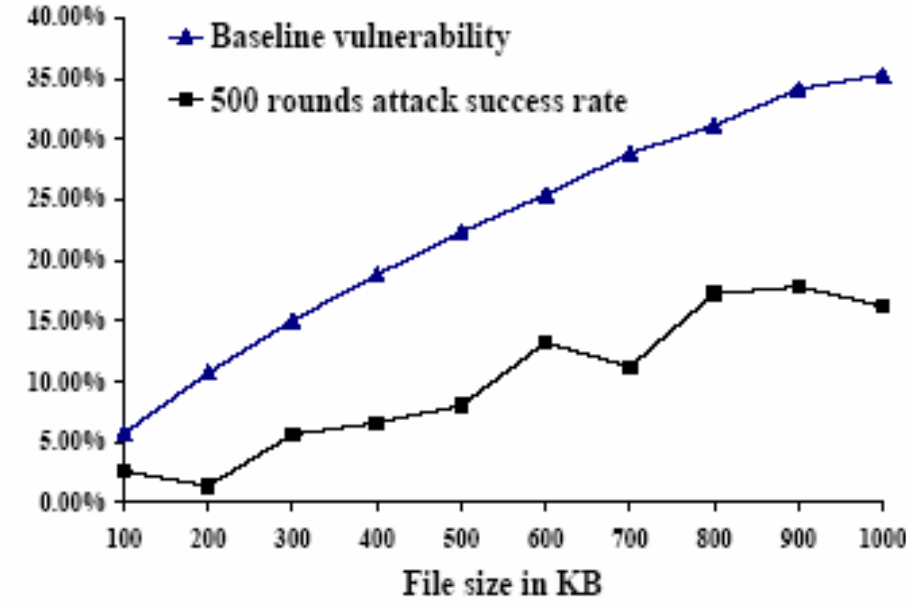


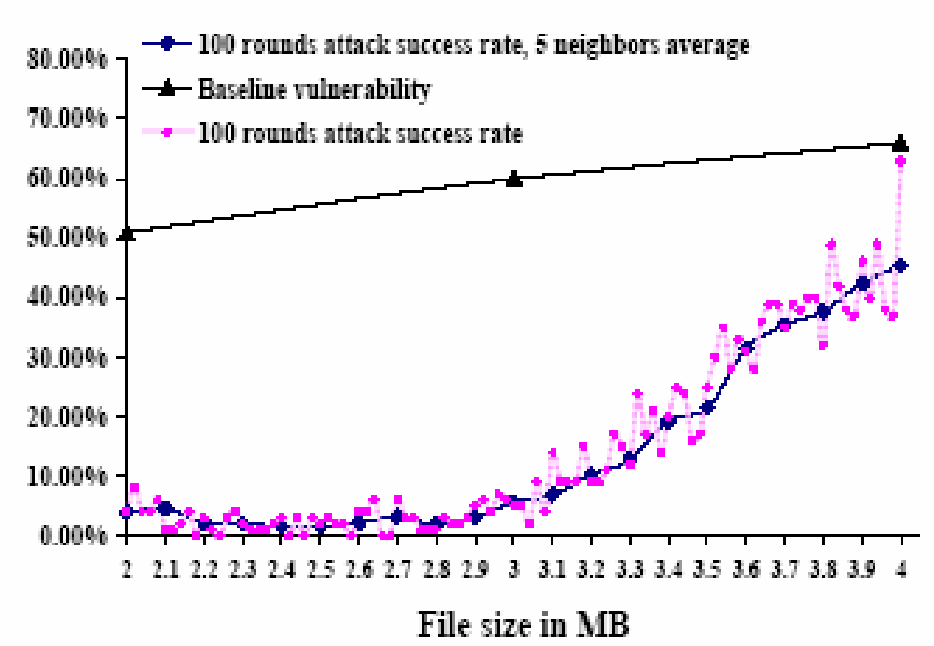


Vulnerability and Save Window Sizes of *vi*

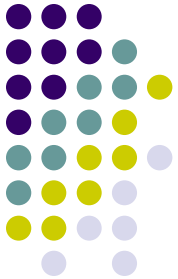
Window of Vulnerability Divided by Save Time, as a Function of File Size

Success Rate of Attacking Vi



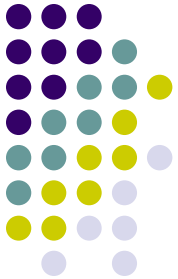


Necessary Conditions of Vi Attack



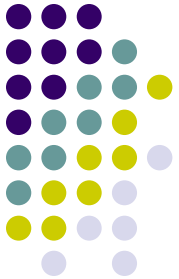
- *vi* has called **open** to create the new file
- *vi* has not called **chown**
- *vi* relinquishes CPU and the attacker is scheduled to run
- the attacker process finishes the file redirection during this run

Reasons for Vi to Relinquish CPU



- Blocked on I/O (file caching effect)
- Preempted because CPU slice is used up
- Preempted by higher priority processes, e.g. *ntpd, kswapd* and *bdflush* kernel threads
- A probability event!

Ongoing work



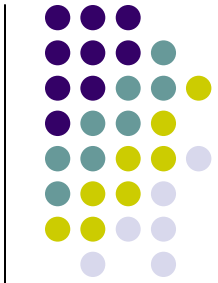
- Prevention of TOCTTOU
- Completeness of CUU model
- Windows file system vulnerabilities

Related Work: Detecting TOCTTOU

- Static analysis
  - Compiler extensions
- Dynamic analysis
  - Dynamic online analysis
    - Eraser
  - Post mortem analysis

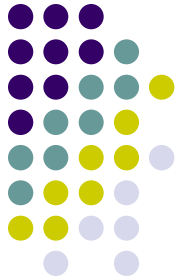
Related Work: Preventing TOCTTOU

- RaceGuard: <**stat, open**>
- Probabilistic approach (Dean and Hu): <**access, open**> ➔ <**access, open**, **access, open**, …>
- Pseudo-transaction (Tsyrklevich and Yee)



Conclusion

- The CUU model of TOCTTOU vulnerabilities
- The dynamic monitoring framework
- The probability and event analysis of attacking *vi*



Thank you!

Questions?