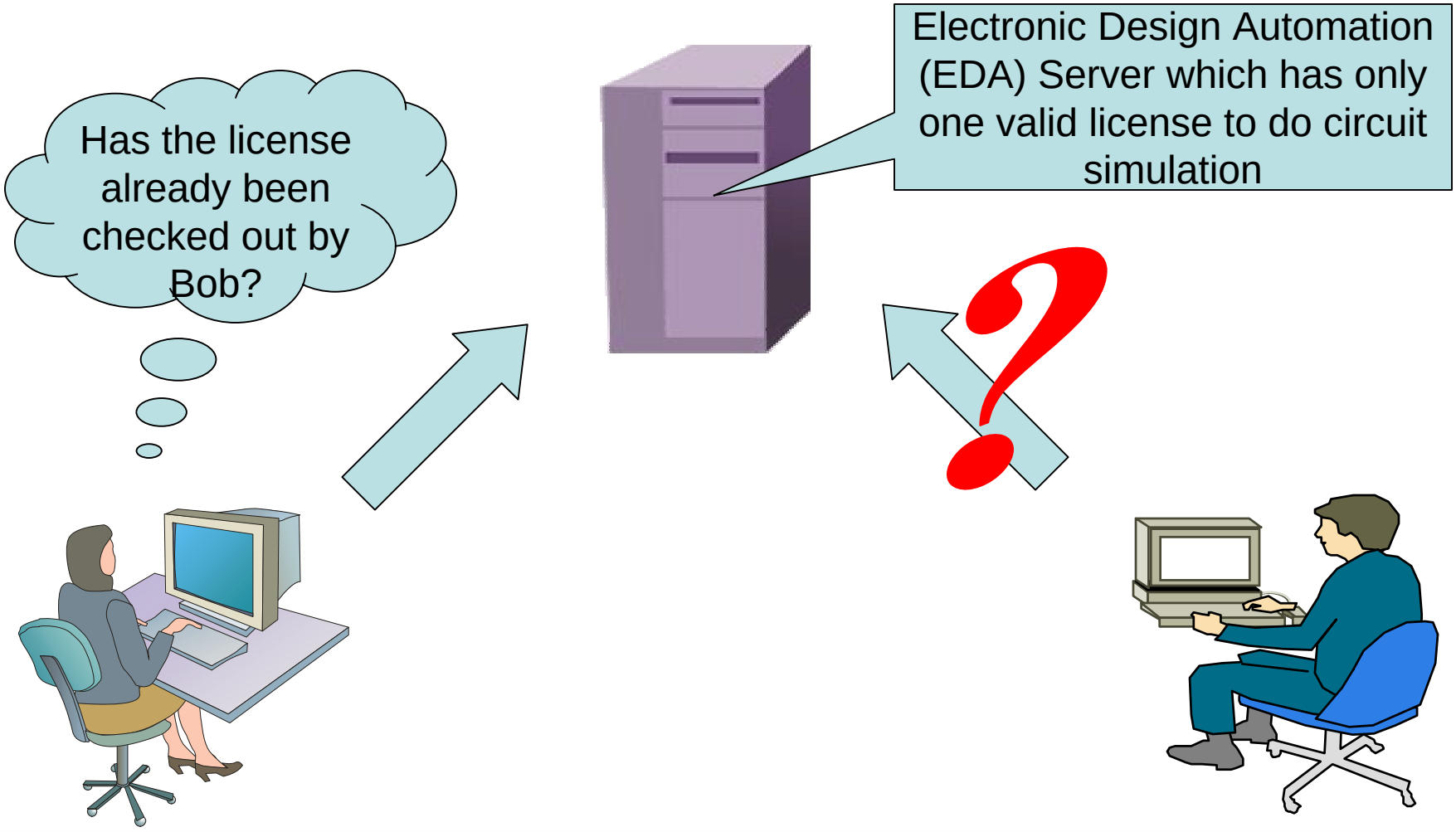
Peeping Tom in the Neighborhood: Keystroke Eavesdropping on Multi-User System

Kehuan Zhang, XiaoFeng Wang
Indiana University

What I will talk about…

- The discovery of a new vulnerability in the shared information on multi-user systems
- An example of attack
  - To infer the key input by the victim

Information Sharing on Multi-user Systems



The output of *top(1)* program

```
top - 07:11:47 up 2 days, 20:38,  2 users,  load average: 0.00, 0.00, 0.00
Tasks: 123 total,   1 running, 122 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.5%us,  0.1%sy,  0.0%ni, 96.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2063204k total,  1564500k used,   498704k free,   186612k buffers
Swap:  3229024k total,        0k used,  3229024k free,   520144k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5929 zkh       20   0  449m 313m  28m S    7 15.5 256:23.03 firefox
 5960 zkh       20   0  147m  77m  36m S    1  3.8  25:08.10 ld-linux.so.2
13985 zkh       20   0  2416 1144  876 R    1  0.1   0:00.04 top
 5523 root      20   0  384m  29m  10m S    0  1.5  15:57.99 Xorg
 5640 zkh       20   0 38732  22m  13m S    0  1.1   4:04.69 gnome-panel
    1 root      20   0  2920 1720  576 S    0  0.1   0:02.46 init
    2 root      15  -5     0    0    0 S    0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S    0  0.0   0:00.02 migration/0
    4 root      15  -5     0    0    0 S    0  0.0   0:01.88 ksoftirqd/0
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/0
    6 root      RT  -5     0    0    0 S    0  0.0   0:00.02 migration/1
    7 root      15  -5     0    0    0 S    0  0.0   0:02.76 ksoftirqd/1
    8 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/1
    9 root      15  -5     0    0    0 S    0  0.0   0:01.96 events/0
   10 root      15  -5     0    0    0 S    0  0.0   0:01.60 events/1
   11 root      15  -5     0    0    0 S    0  0.0   0:00.02 khelper
   51 root      15  -5     0    0    0 S    0  0.0   0:00.00 kintegrityd/0
```

The procfs (process file system)

- A pseudo file system
- Provides information about the system and each process

System information provided by *procfs*

```
1       210     4764    5579    57      7836    filesystems     sched_debug
10      211     4765    5605    5702    7837    fs              schedstat
10086   2228    4768    5608    5703    7838    interrupts      scsi
11      2404    4769    5609    5705    7850    iomem           self
1263    2951    4851    5612    5707    7854    ioports         slabinfo
1264    3       4862    5613    5709    7997    irq             stat
127     3980    4875    5615    5710    8       kallsyms        swaps
131     4       4916    5621    5712    9       kcore           sys
1336    4365    4936    5626    5718    acpi    key-users       sysrq-trigger
1337    4366    4938    5628    5721    asound  kmsg            sysvipc
1338    4368    4939    5631    5726    buddyinfo       kpagecount      timer_list
13927   4369    4977    5633    5731    bus     kpageflags      timer_stats
13934   4370    5       5635    5791    cgroups latency_stats   tty
13936   4420    5007    5640    5792    cmdline loadavg         uptime
14068   4443    51      5643    5793    cpuinfo locks           version
168     4445    5104    5646    5796    crypto  meminfo         version_signature
169     4468    5182    5662    58      devices misc            vmallocinfo
170     4490    52      5668    5927    diskstats       modules         vmcore
2       4491    54      5670    5928    dma     mounts          vmstat
2013    4520    55      5673    5929    dri     mtrr            zoneinfo
2014    4566    5505    5676    5960    driver  net
2069    4675    5522    5678    6       execdomains     pagetypeinfo
2070    4742    5523    5695    7       fb      partitions
```

File contents in /proc/interrupts

```
           CPU0       CPU1
  0:       1072          0   IO-APIC-edge      timer
  1:          2          0   IO-APIC-edge      i8042
  4:          3          0   IO-APIC-edge
  7:          0          0   IO-APIC-edge      parport0
  8:         53          0   IO-APIC-edge      rtc0
  9:          0          0   IO-APIC-fasteoi   acpi
 12:          4          0   IO-APIC-edge      i8042
 14:     474360          0   IO-APIC-edge      ata_piix
 15:          0          0   IO-APIC-edge      ata_piix
 16:    1674852          0   IO-APIC-fasteoi   eth0, i915@pci:0000:00:02.0
 18:          0          0   IO-APIC-fasteoi   uhci_hcd:usb4
 20:     329344          0   IO-APIC-fasteoi   ata_piix
 21:          6          0   IO-APIC-fasteoi   ehci_hcd:usb1, uhci_hcd:usb2
 22:       1044          0   IO-APIC-fasteoi   uhci_hcd:usb3
 23:      56253          0   IO-APIC-fasteoi   uhci_hcd:usb5, Intel ICH7
NMI:          0          0   Non-maskable interrupts
LOC:   30046493   33508704   Local timer interrupts
RES:    4084892    2899075   Rescheduling interrupts
CAL:      75109     122026   function call interrupts
TLB:      41597      40468   TLB shootdowns
SPU:          0          0   Spurious interrupts
ERR:          0
MIS:          0
```

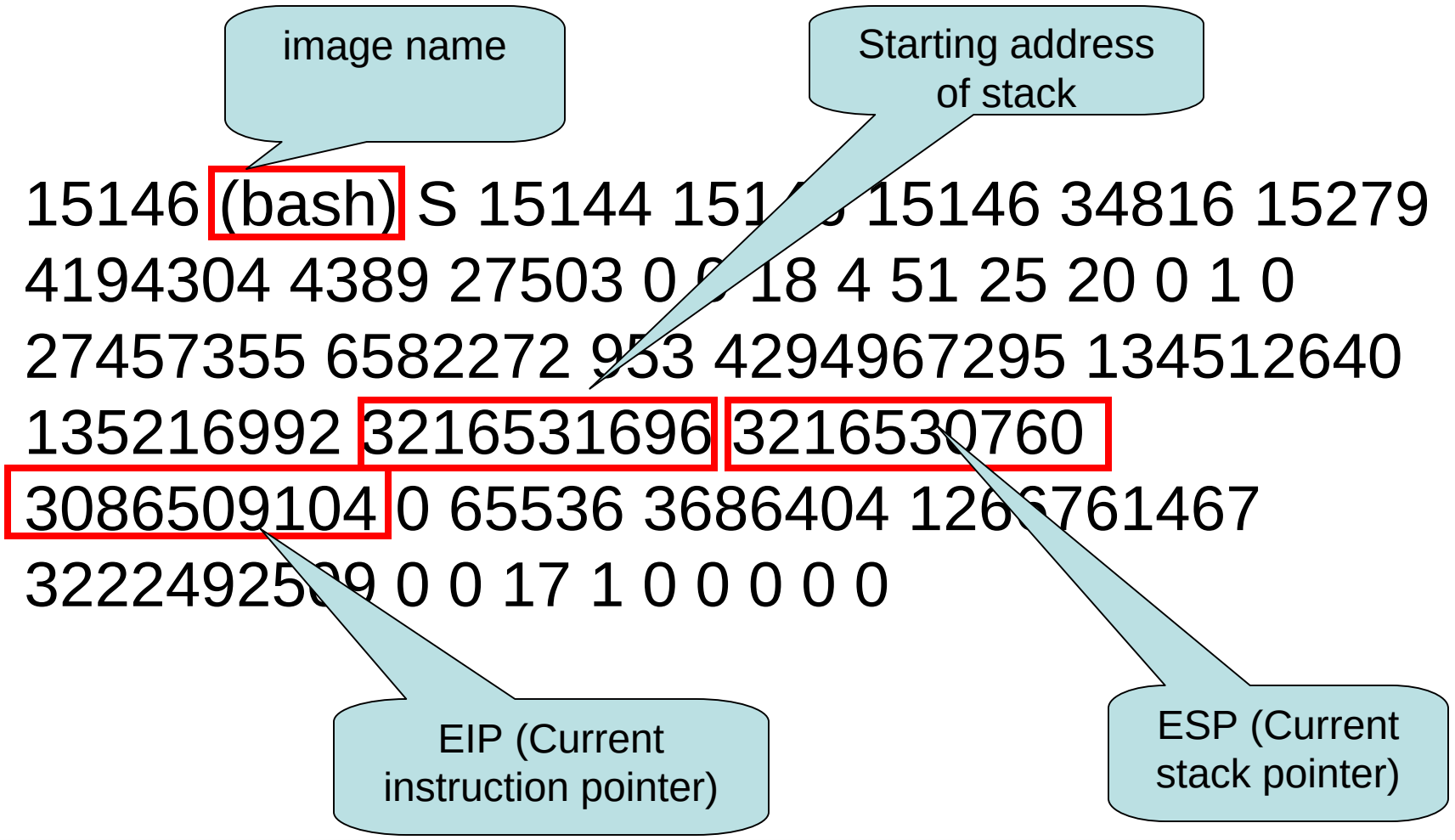
File contents in /proc/slabinfo

```
sock_inode_cache     573    588    384   21    2 : tunables    0    0    0 :
file_lock_cache       78     78    104   39    1 : tunables    0    0    0 :
Acpi-Operand        1419   1428     40  102    1 : tunables    0    0    0 :
taskstats             48     48    328   24    2 : tunables    0    0    0 :
page_cgroup       335578 339490     24  170    1 : tunables    0    0    0 :
proc_inode_cache    1585   1628    368   22    2 : tunables    0    0    0 :
sigqueue              56     56    144   28    1 : tunables    0    0    0 :
radix_tree_node    10104  10152    296   27    2 : tunables    0    0    0 :
bdev_cache            32     32    512   16    2 : tunables    0    0    0 :
sysfs_dir_cache    13084  13090     48   85    1 : tunables    0    0    0 :
inode_cache         8294   8326    344   23    2 : tunables    0    0    0 :
dentry            784649 784650    136   30    1 : tunables    0    0    0 :
buffer_head        64014  64192     64   64    1 : tunables    0    0    0 :
vm_area_struct     13318  13800     88   46    1 : tunables    0    0    0 :
files_cache          117    126    384   21    2 : tunables    0    0    0 :
signal_cache         161    175    640   25    4 : tunables    0    0    0 :
sighand_cache        155    161   1408   23    8 : tunables    0    0    0 :
task_struct          228    250   3216   10    8 : tunables    0    0    0 :
anon_vma            5109   5376     16  256    1 : tunables    0    0    0 :
idr_layer_cache      619    624    152   26    1 : tunables    0    0    0 :
kmalloc-4096          44     56   4096    8    8 : tunables    0    0    0 :
kmalloc-2048         289    384   2048   16    8 : tunables    0    0    0 :
slabinfo [RO]                                          42,1           71%
```

Process-related information

```
zkh@zkh-iu-office:/proc/13936$ ps
13936 pts/0    00:00:00 bash
13996 pts/0    00:00:00 ps
zkh@zkh-iu-office:/proc/13936$ ls
attr              cpuset   io         mountinfo   pagemap   stat
auxv              cwd      latency    mounts      root      statm
cgroup            environ  limits     mountstats  sched     status
clear_refs        exe      loginuid   net         schedstat task
cmdline           fd       maps       oom_adj     sessionid wchan
coredump_filter   fdinfo   mem        oom_score   smaps
```

Content of /proc/<pid>/stat



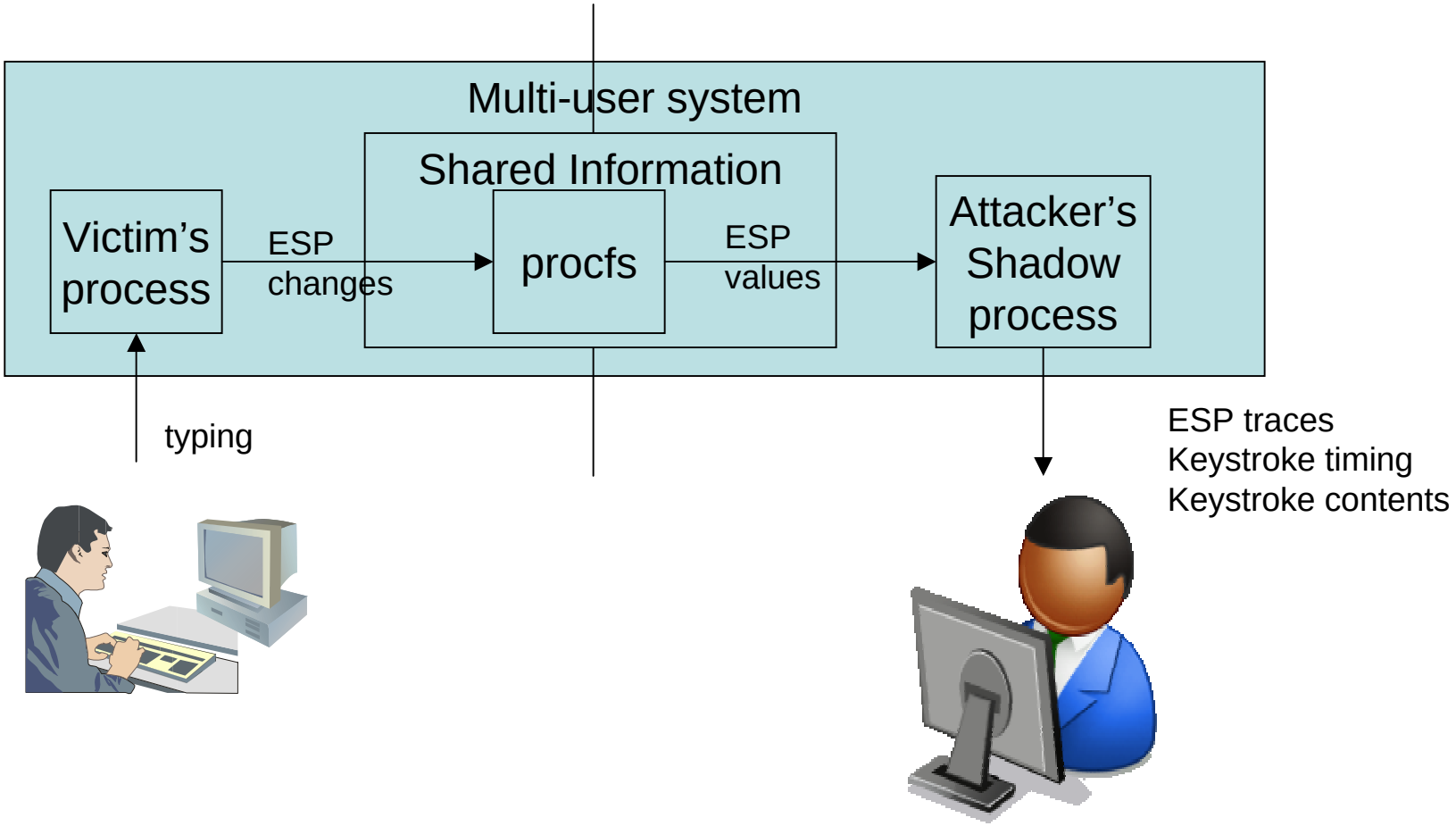
Contents of /proc/<pid>/maps

```
08048000-080f4000 r-xp 00000000 08:01 344069     /bin/bash
080f4000-080f9000 rw-p 000ac000 08:01 344069     /bin/bash
080f9000-080fe000 rw-p 080f9000 00:00 0
09856000-09ab5000 rw-p 09856000 00:00 0          [heap]
b7c5b000-b7c65000 r-xp 00000000 08:01 3441330    /lib/tls/i686/cmov/libnss_fi
b7c65000-b7c66000 r--p 00009000 08:01 3441330    /lib/tls/i686/cmov/libnss_fi
b7c66000-b7c67000 rw-p 0000a000 08:01 3441330    /lib/tls/i686/cmov/libnss_fi
b7c67000-b7c70000 r-xp 00000000 08:01 3441332    /lib/tls/i686/cmov/libnss_ni
b7c70000-b7c71000 r--p 00008000 08:01 3441332    /lib/tls/i686/cmov/libnss_ni
b7c71000-b7c72000 rw-p 00009000 08:01 3441332    /lib/tls/i686/cmov/libnss_ni
b7c72000-b7c87000 r-xp 00000000 08:01 3441327    /lib/tls/i686/cmov/libnsl-2.
b7c87000-b7c88000 r--p 00014000 08:01 3441327    /lib/tls/i686/cmov/libnsl-2.
b7c88000-b7c89000 rw-p 00015000 08:01 3441327    /lib/tls/i686/cmov/libnsl-2.
b7c89000-b7c8b000 rw-p b7c89000 00:00 0
b7c8b000-b7c92000 r-xp 00000000 08:01 3441328    /lib/tls/i686/cmov/libnss_co
b7c92000-b7c93000 r--p 00006000 08:01 3441328    /lib/tls/i686/cmov/libnss_co
b7c93000-b7c94000 rw-p 00007000 08:01 3441328    /lib/tls/i686/cmov/libnss_co
b7ca5000-b7ce4000 r--p 00000000 08:01 1754042    /usr/lib/locale/en_US.utf8/L
b7ce4000-b7dc5000 r--p 00000000 08:01 1754041    /usr/lib/locale/en_US.utf8/L
b7dc5000-b7dc6000 rw-p b7dc5000 00:00 0
b7dc6000-b7f1e000 r-xp 00000000 08:01 3441321    /lib/tls/i686/cmov/libc-2.8.
b7f1e000-b7f20000 r--p 00158000 08:01 3441321    /lib/tls/i686/cmov/libc-2.8.
maps [RO]                                                    1,1           Top
:set nowrap
```

Privacy Implications

- All the information mentioned is globally readable
- Key inference (this paper)
- Others
  - Trace program execution path
  - Infer user behavior or input data
  - ASLR (Address Space Layout Randomization)

What we did – keystroke eavesdropping



Related work

- Keystroke inference has been well studied
  - Ssh traffic [SWT 2001]
  - Keyboard acoustic [AA 2004, ZZT 2005,BWY 2006]
  - Electromagnetic emanation [VP 2009]
- What is different
  - A new and more accurate way to get keystroke timing
  - Use semantic information to get multiple timing sequences
  - Novel technology to get better results from multiple sequences

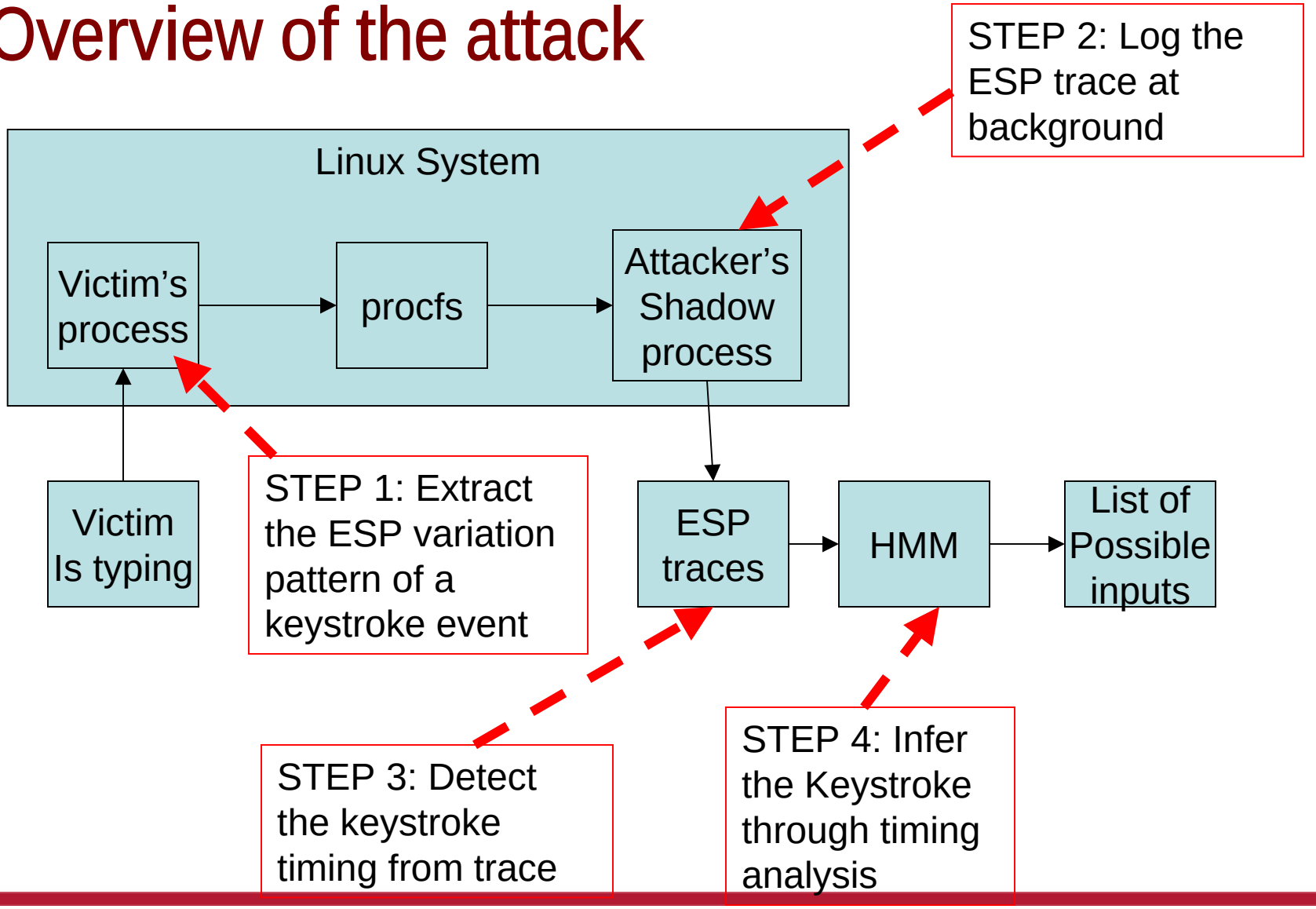
Contributions

- Discover a new vulnerability in process file system
- Develop techniques to exploit this vulnerability to determine keystroke timing
- Augment the keystroke analysis technique with semantic information

Assumptions

- Multi-user system
- Capability to execute programs
- Multi-core CPU

Overview of the attack



Step 1: Extract ESP patterns of keystrokes

- Consists of ESP value when a system call is made
  - System call is time consuming
  - Can be reliably captured by the shadow program
  - Most ESP values captured are belonging to system call
- Two types of program
  - Deterministic
  - Non-deterministic

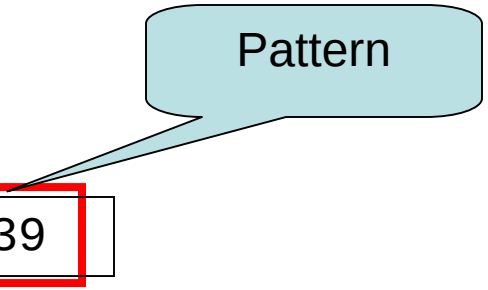
Extract pattern for deterministic programs

- Examples: Vim, sshd
- Use differential analysis technique
- Modify *strace* source code to output the ESP value when a system call is made
- Example
  - First run without any keystroke

```
100 120 108 112
```

  - Second run with a single keystroke

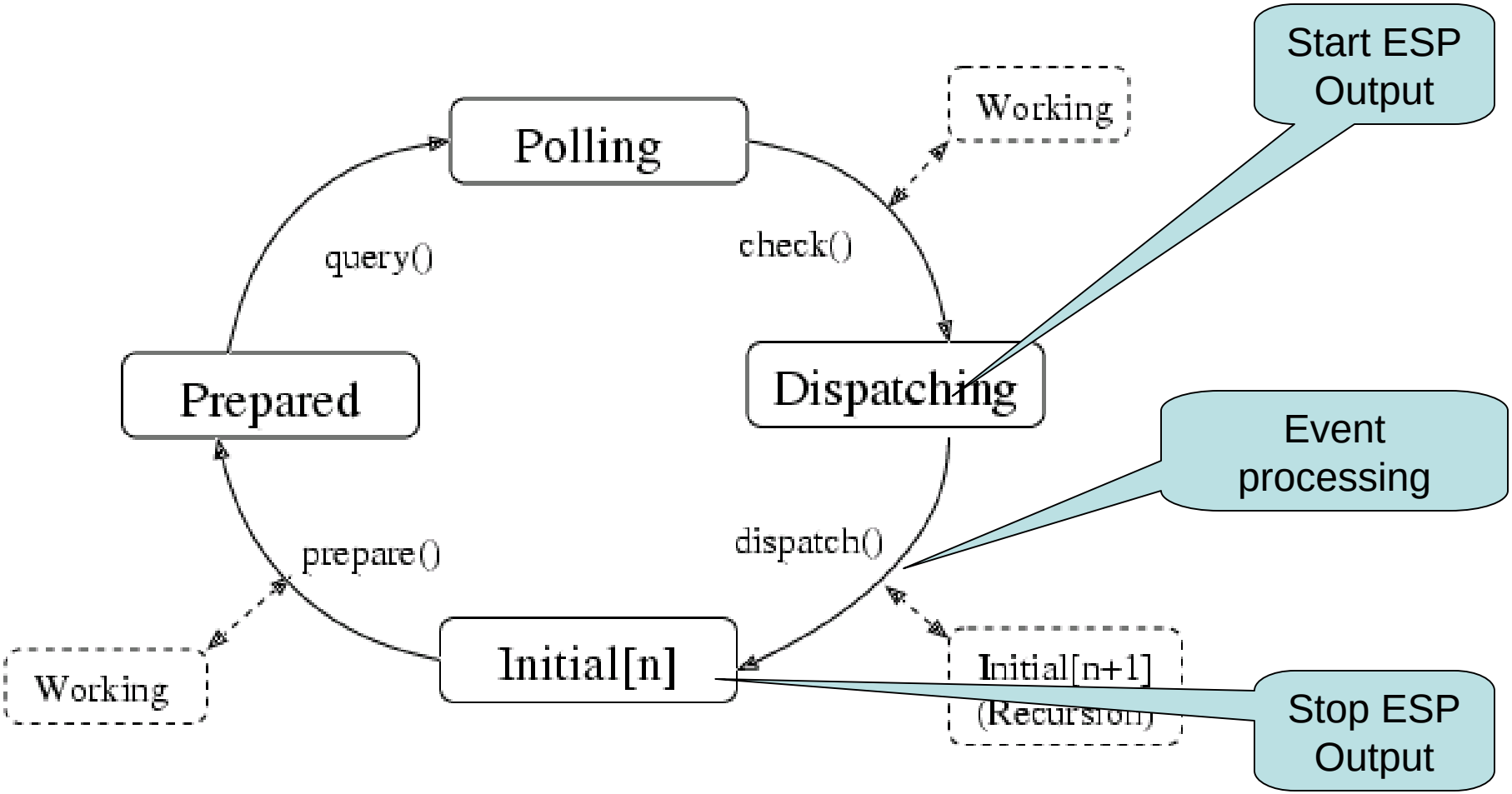
```
100 120 108 112 103 119 122 139
```



Extract pattern for non-deterministic programs

- Examples: Gedit
  - Driven by random Events
    - like cursor blinks, screen refresh, buffer flush
- Solution: Use instruction level analysis

Example: GTK+ Event loop in Gedit



Step 2: Log ESP traces

- Using a shadow process
- Make it efficient as much as possible
  - Written in assembly language
  - Each sample has only two system calls: lseek() and read()
  - Use a buffer to reduce cost of writing data to disk

Step 3: Determine keystroke timings

- Challenge:
  - To recognize a keystroke event from an incomplete ESP trace
- Solution
  - Convert into a LCS (longest common subsequence) problem
    - Let X be “ABCDEFG" and Y be “BGCEHAF". The longest common subsequence between X and Y is “BCEF".
    - We regard each ESP value as a unique character
    - Using existed dynamic programming algorithms

Examples: Determine keystroke timings

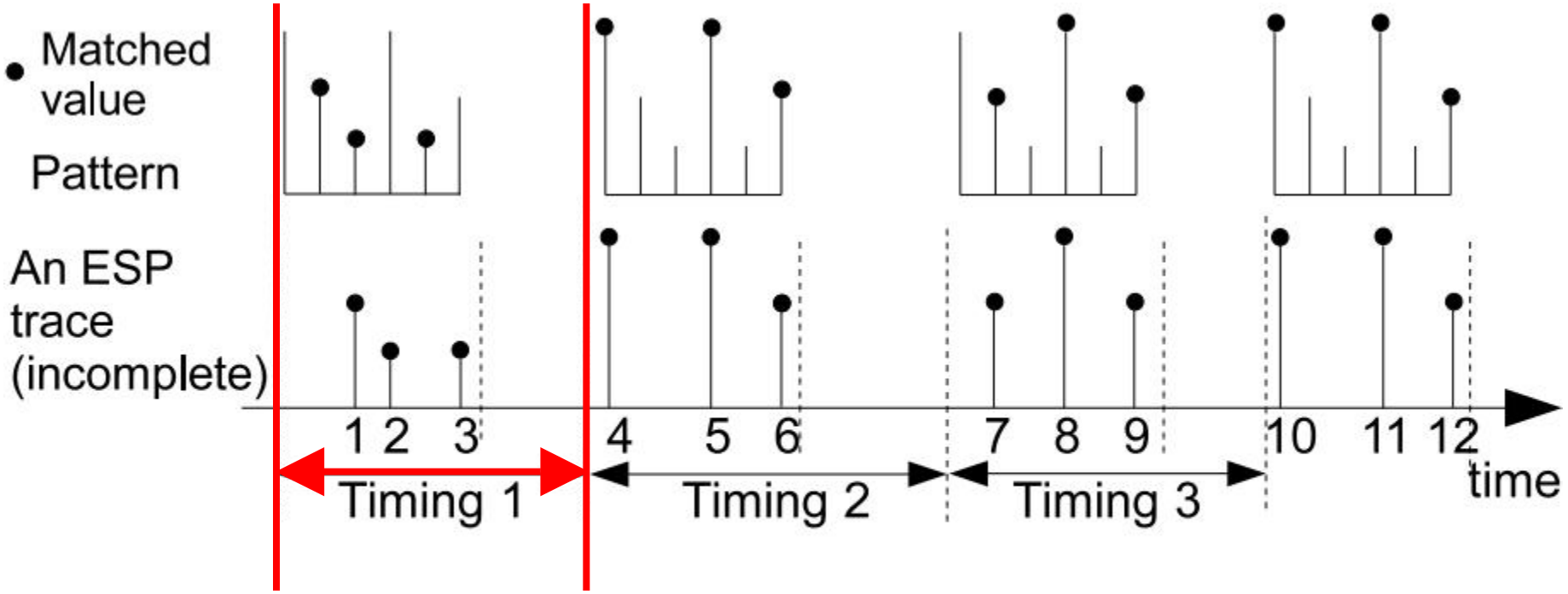
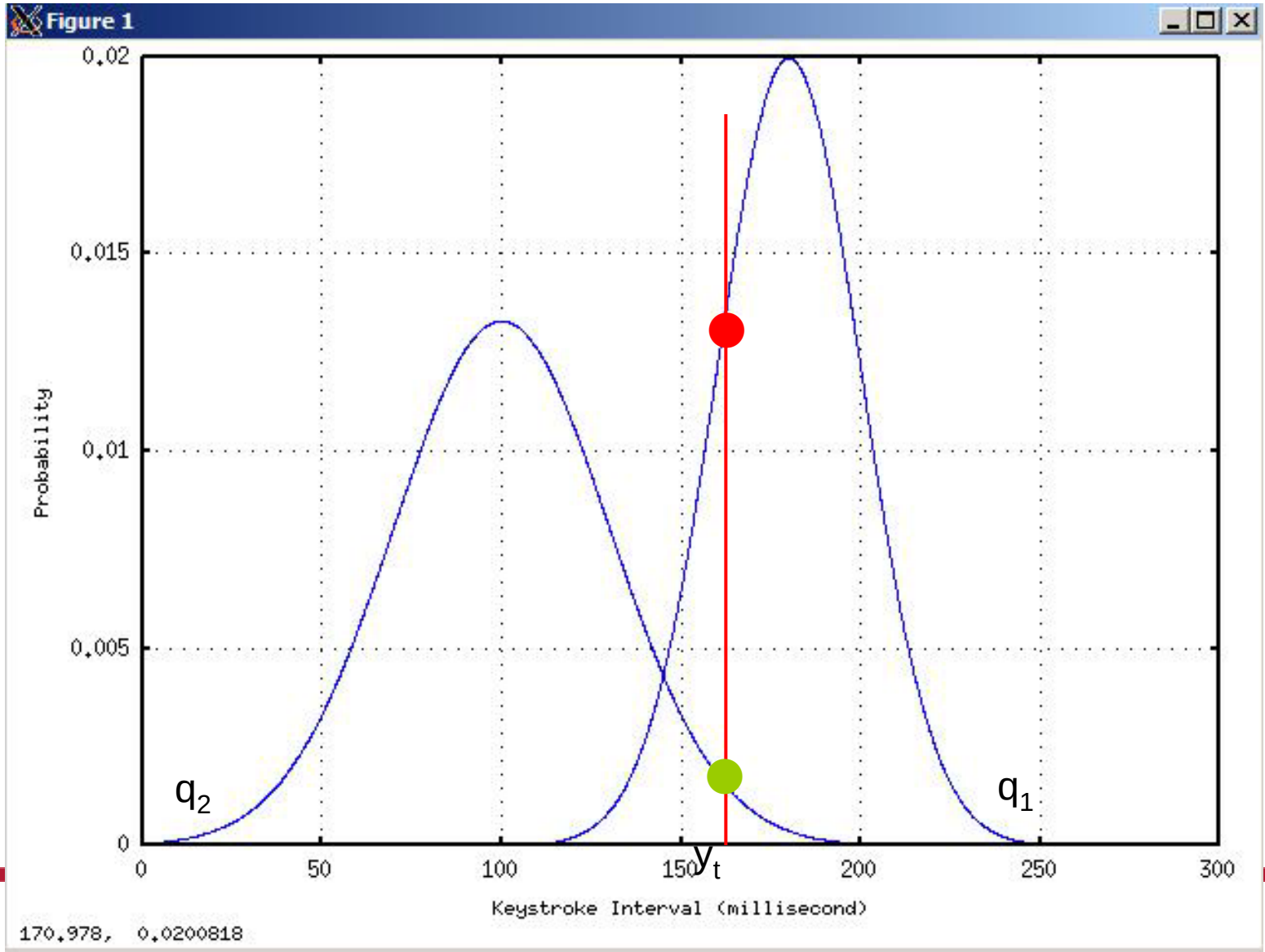


Figure 5: Pattern matching on an ESP trace and the timing interval

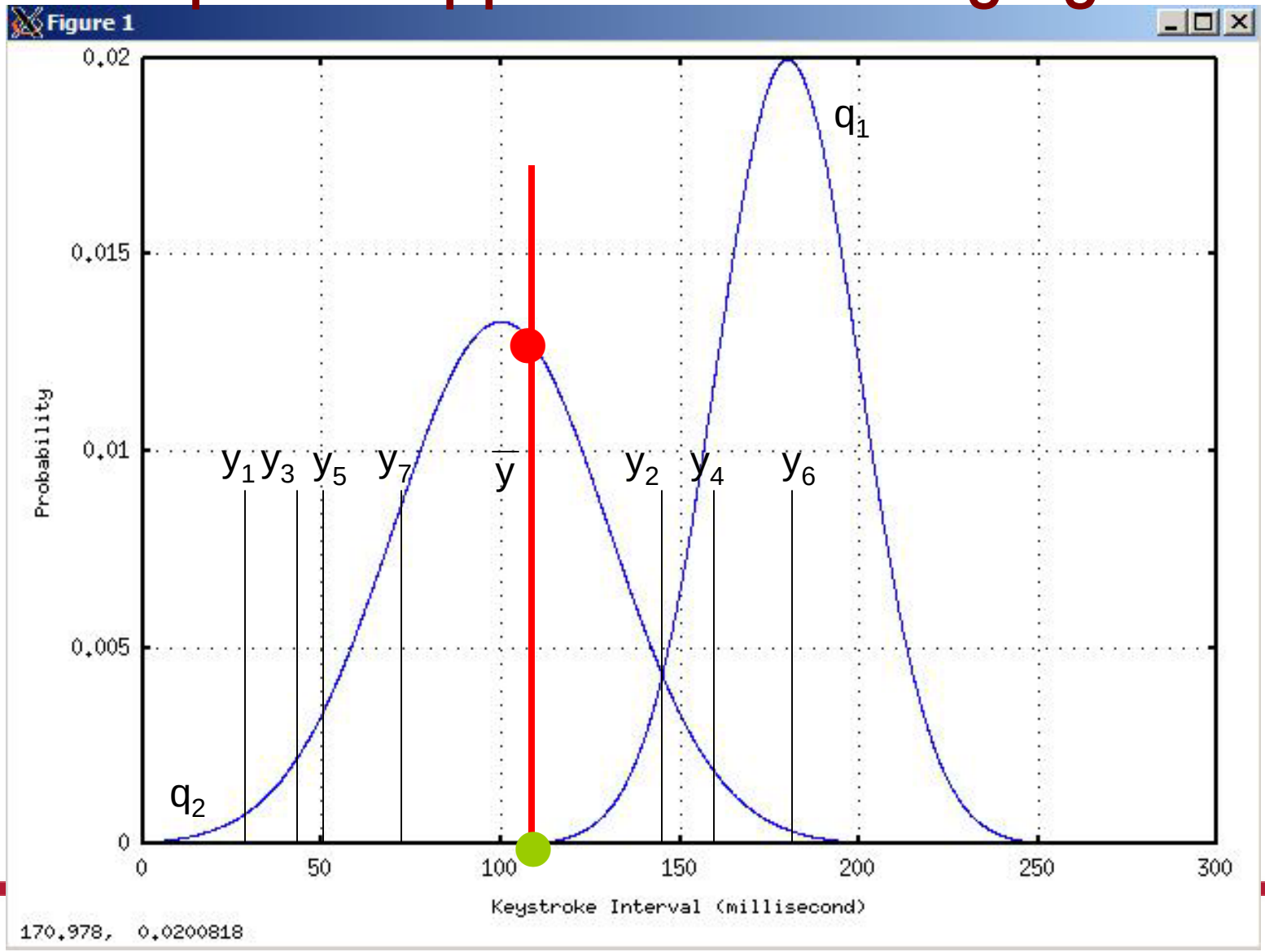
Step 4: Key Inferences

- Use the Hidden Markov Model (HMM)
- Based on the n-Viterbi algorithm
- Our contribution:
  - Leverage the information from multiple timing sequences

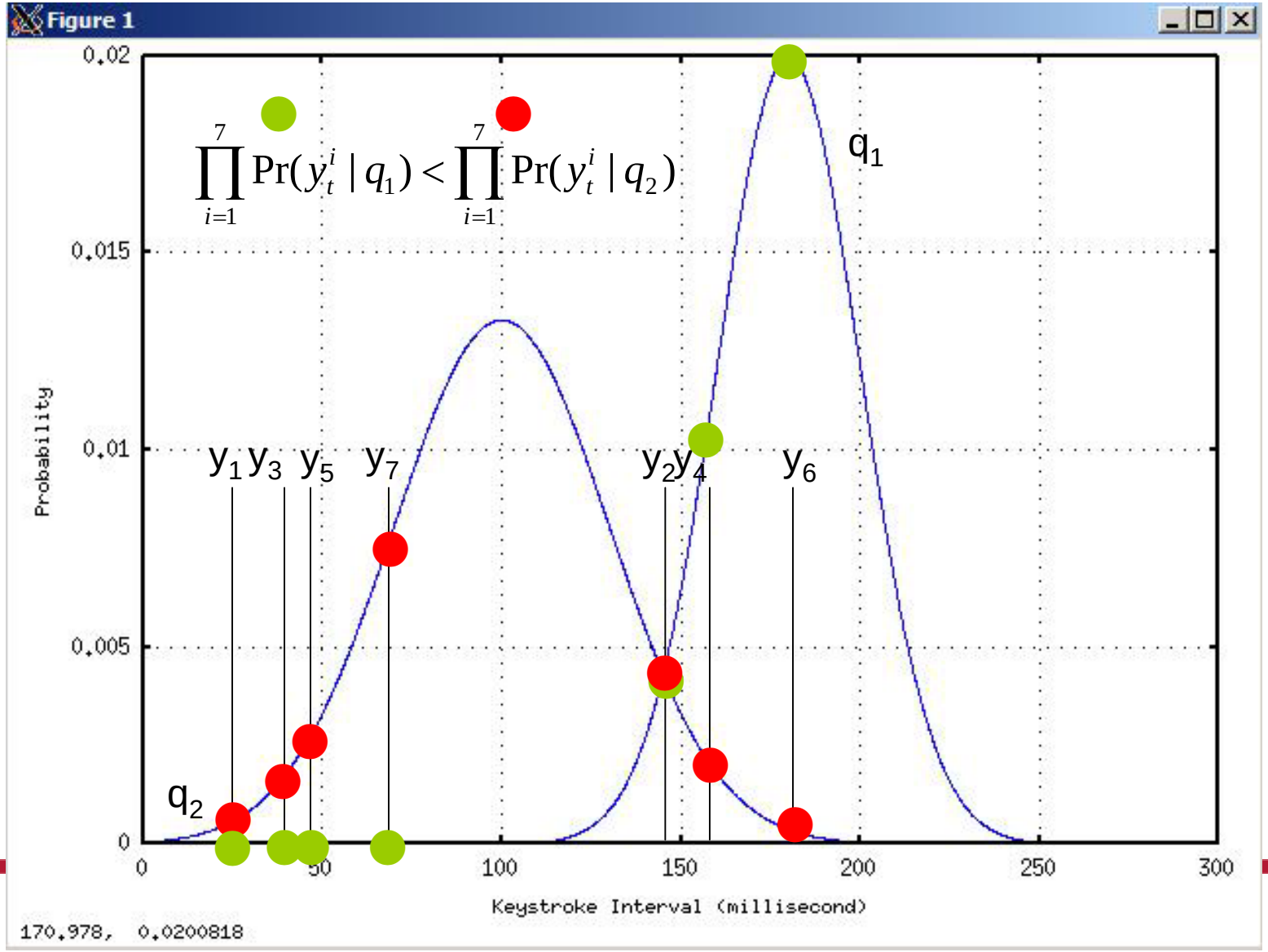
Limitation of single timing sequence



Multi-sequence approach 1: Averaging



Multi-sequence approach 2: Combined probability



Evaluations

- Things to be evaluated
  - The accuracy of the keystroke timing detected
  - The performance of key sequence inference
- Three sample applications
  - Vim
  - sshd
  - Gedit
- Platform
  - Intel Core 2 Duo E6700, 3GB RAM
  - Red Hat Linux Enterprise 4

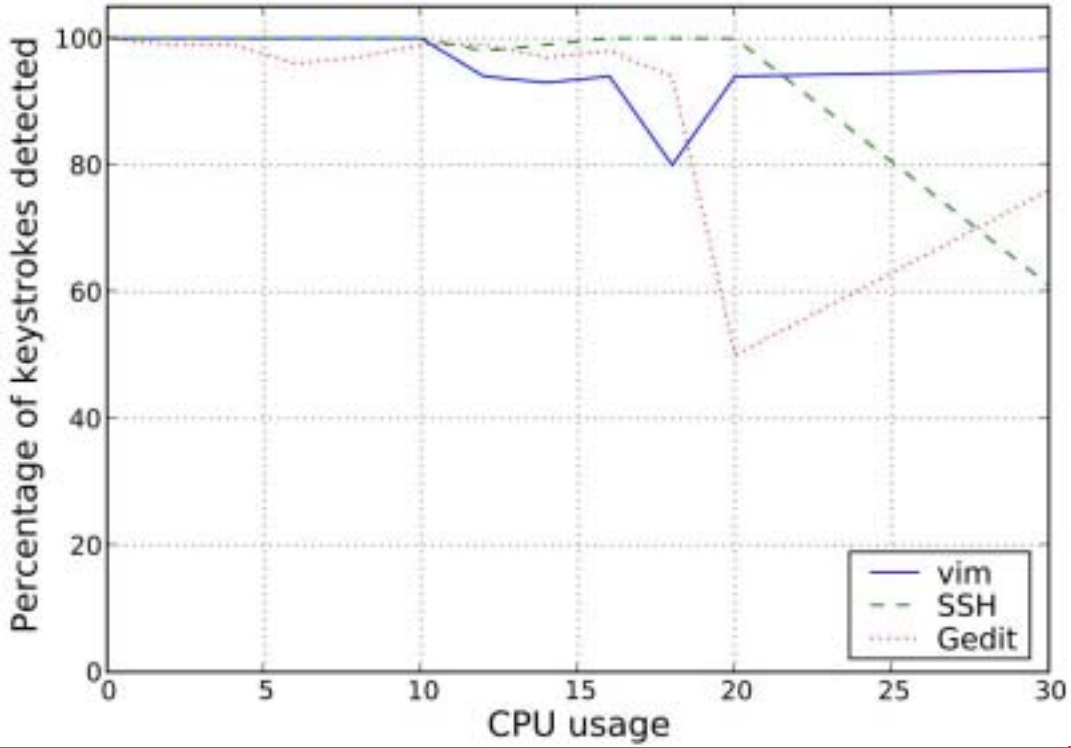
Timing accuracy

Table 3: Examples of the timings measured from ESP traces (Measured) and the real timings (Real) in milliseconds.

Timings	vim		ssh		Gedit	
	measured	real	measured	real	measured	real
1	80	81	135	135	301	303
2	139	139	124	123	285	285
3	88	88	103	103	259	259
4	101	101	110	109	236	236
5	334	335	134	134	181	182
6	86	87	111	110	265	265
7	124	124	132	132	174	174

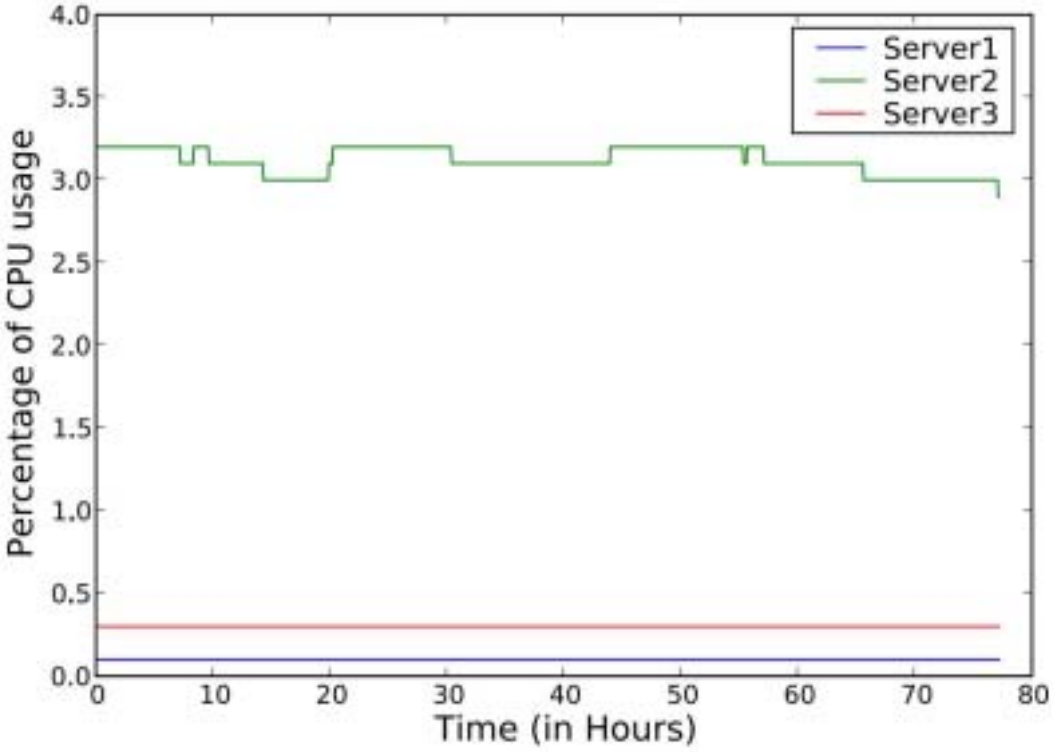
Impacts of server workloads

- Percentage of keystrokes detected vs. CPU usage



Impacts of server workloads

- CPU usage history of three real-world severs within 72 hours



Evaluation of password inference

- Acquire the training data for HMM
- Generate three passwords randomly
- Get 50 timing sequences for each password
- Run HMM to generate a list of guessed passwords
- Identify the position of the real password on the list
- Calculate the percentage of the search space

Key Inference -- password

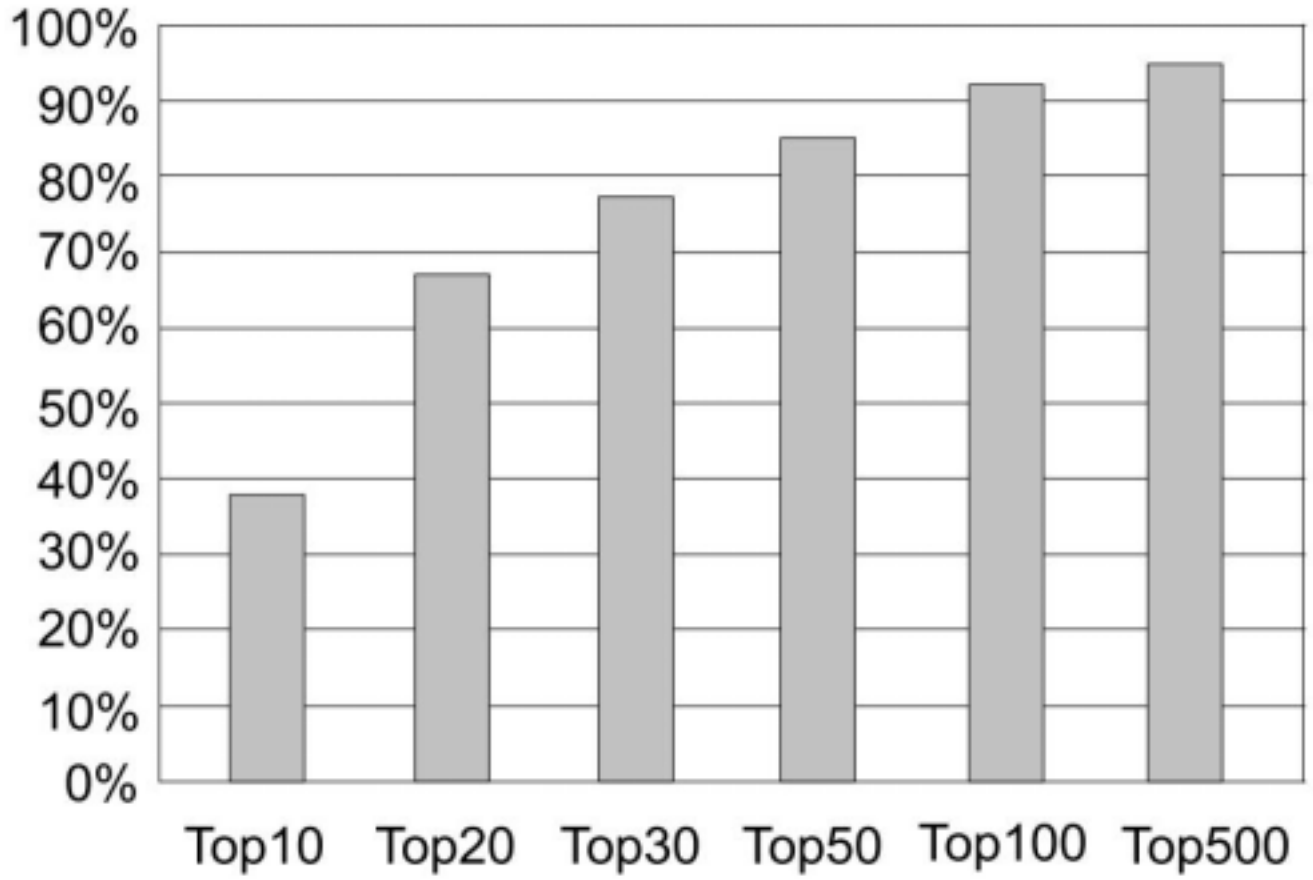
Table 4: The percentage of the search space the attacker has to search before the right password is found.

Method	Test Cases		
	password 1	password 2	password 3
Baseline(n-Viterbi)	7.8%	6.6%	6.8%
Timing Averaging	0.38%	0.34%	0.05%
m-n-Viterbi	0.39%	0.34%	0.05%

Evaluation of English word inference

- Build a dictionary
- Drawn words from the dictionary randomly
- Get a keystroke timing for each word
- Ask HMM to output a list of guessed words
- Delete invalid output which is not appear in the dictionary
- Find the position and calculate the percentage

Key Inference – English words



Discussion

- Other UNIX-like systems
  - No ESP/EIP information
  - But provide CPU time information for system calls
- Defense
  - Patch the kernel
  - Need a complete evaluation about the information leakage

Conclusion

- Privacy vulnerability in the process file system
  - Keystroke inference attack
- Further study
  - Other attacks
  - Potential mitigation technology

Q & A