Software Security: Defenses & Principles

CS 161: Computer Security

Prof. Vern Paxson

TAs: Devdatta Akhawe, Mobin Javed & Matthias Vallentin

http://inst.eecs.berkeley.edu/~cs161/

January 25, 2011

Testing for Software Security Issues

- What makes testing a program for security problems difficult?
  - We need to test for the *absence* of something
    - Security is a negative property!
      - "nothing bad happens, even in really unusual circumstances"
  - Normal inputs rarely stress security-vulnerable code
- How can we test more thoroughly?

Testing for Software Security Issues

- What makes testing a program for security problems difficult?
  - We need to test for the absence of something
    - Security is a negative property!
      - "nothing bad happens, even in really unusual circumstances"
  - Normal inputs rarely stress security-vulnerable code
- How can we test more thoroughly?
  - Random inputs (*fuzz testing*)

Testing for Software Security Issues

- What makes testing a program for security problems difficult?
  - We need to test for the absence of something
    - Security is a negative property!
      - "nothing bad happens, even in really unusual circumstances"
  - Normal inputs rarely stress security-vulnerable code
- How can we test more thoroughly?
  - Random inputs (*fuzz testing*)
  - Mutation
  - Spec-driven
- How do we tell when we've found a problem?
  - Crash or other deviant behavior; enable expensive checks
- How do we tell that we've tested enough?
  - **Hard**: but *code coverage* tools can help

```
int deref(int *p) {
    return *p;
}
```

```
/* requires: p != NULL
             (and p a valid pointer) */
int deref(int *p) {
    return *p;
}
```

```
int sum(int a[], size_t n) {
  int total = 0;
  for (size_t i=0; i<n; i++)
    total += a[i];
  return total;
}
```

```
/* requires: a != NULL && size(a) >= n */
int sum(int a[], size_t n) {
  int total = 0;
  for (size_t i=0; i<n; i++)
    total += a[i];
  return total;
}
```

```
/* requires: a != NULL && size(a) >= n */
int sum(int a[], size_t n) {
  int total = 0;
  for (size_t i=0; i<n; i++)

    total += a[i];
  return total;
}
```

```
/* requires: a != NULL && size(a) >= n */
int sum(int a[], size_t n) {
  int total = 0;
  for (size_t i=0; i<n; i++)
      /* 0 <= i && i < n && n <= size(a) */
    total += a[i];
  return total;
}
```

```
int sumderef(int *a[], size_t n) {
    int total = 0;
    for (size_t i=0; i<n; i++)
         total += *(a[i]);
    return total;
}
```

```
/* requires: a != NULL &&
     size(a) >= n &&
           ???                                  */
int sumderef(int *a[], size_t n) {
    int total = 0;
    for (size_t i=0; i<n; i++)
         total += *(a[i]);
    return total;
}
```

```
/* requires: a != NULL &&
     size(a) >= n &&
     for all j in 0..n-1, a[j] != NULL */
int sumderef(int *a[], size_t n) {
    int total = 0;
    for (size_t i=0; i<n; i++)
        total += *(a[i]);
    return total;
}
```

```
void *mymalloc(size_t n) {
    void *p = malloc(n);
    if (!p) { perror("malloc"); exit(1); }
    return p;
}
```

```
/* ensures: retval != NULL */
void *mymalloc(size_t n) {
    void *p = malloc(n);
    if (!p) { perror("malloc"); exit(1); }
    return p;
}
```

```
char *tbl[N];


int hash(char *s) {
  int h = 17;
  while (*s)
    h = 257*h + (*s++) + 3;
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;
  while (*s)
    h = 257*h + (*s++) + 3;
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                    /* 0 <= h */
  while (*s)
    h = 257*h + (*s++) + 3;
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                    /* 0 <= h */
  while (*s)                     /* 0 <= h */
    h = 257*h + (*s++) + 3;
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                          /* 0 <= h */
  while (*s)                           /* 0 <= h */
    h = 257*h + (*s++) + 3;            /* 0 <= h */
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                          /* 0 <= h */
  while (*s)                           /* 0 <= h */
    h = 257*h + (*s++) + 3;            /* 0 <= h */
  return h % N; /* 0 <= retval < N */
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                          /* 0 <= h */
  while (*s)                           /* 0 <= h */
    h = 257*h + (*s++) + 3;            /* 0 <= h */
  return h % N; /* 0 <= retval < N */
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                          /* 0 <= h */
  while (*s)                           /* 0 <= h */
    h = 257*h + (*s++) + 3;            /* 0 <= h */
  return h % N; /* 0 <= retval < N */
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                          /* 0 <= h */
  while (*s)                           /* 0 <= h */
    h = 257*h + (*s++) + 3;            /* 0 <= h */
  return h % N; /* 0 <= retval < N */
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

Fix?

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
unsigned int hash(char *s) {
  unsigned int h = 17;          /* 0 <= h */
  while (*s)                    /* 0 <= h */
    h = 257*h + (*s++) + 3;     /* 0 <= h */
  return h % N; /* 0 <= retval < N */
}

bool search(char *s) {
  unsigned int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

5 Minute Break

Questions Before We Proceed?

TL-15

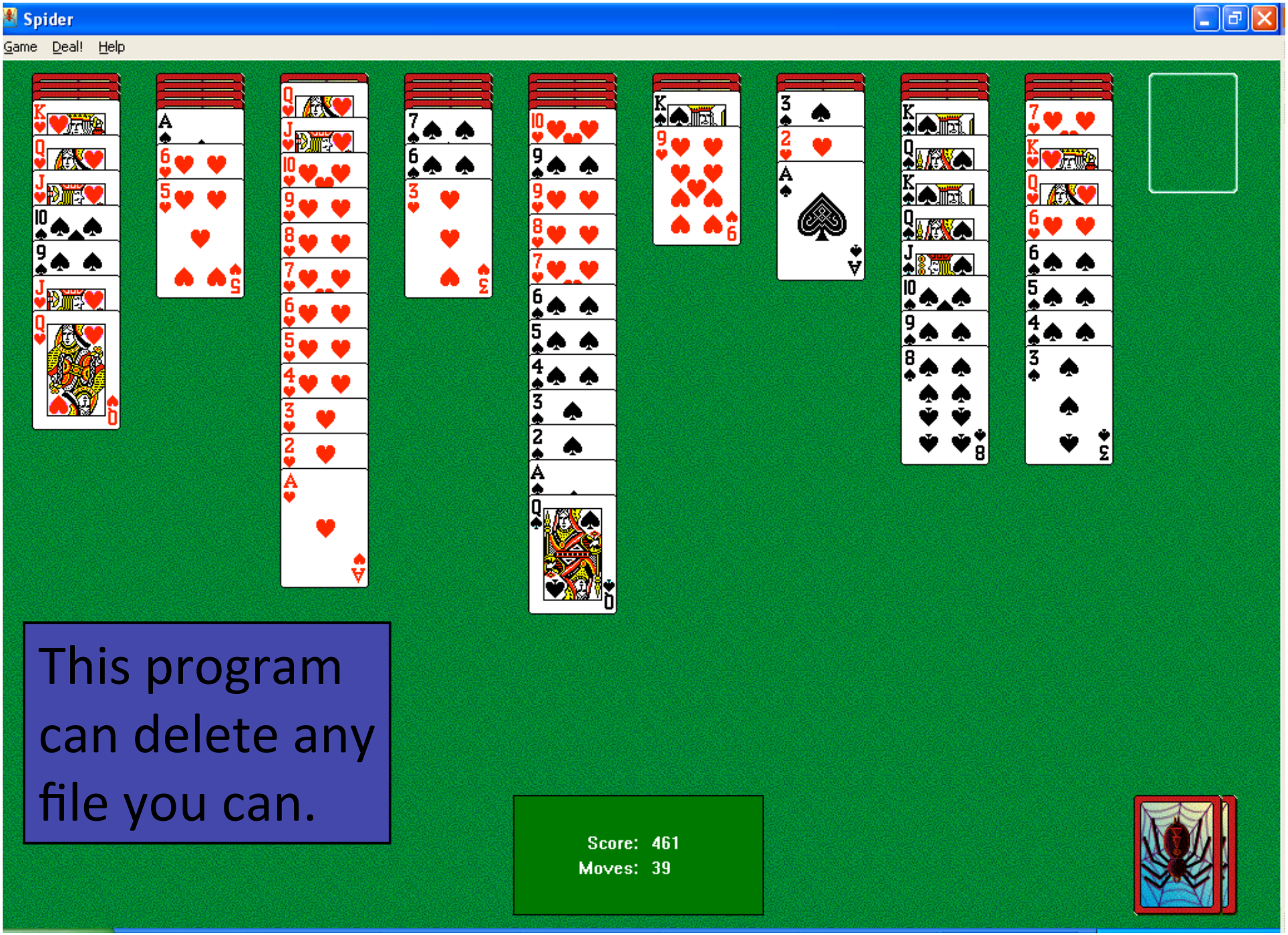
TL-30

TRTL-30

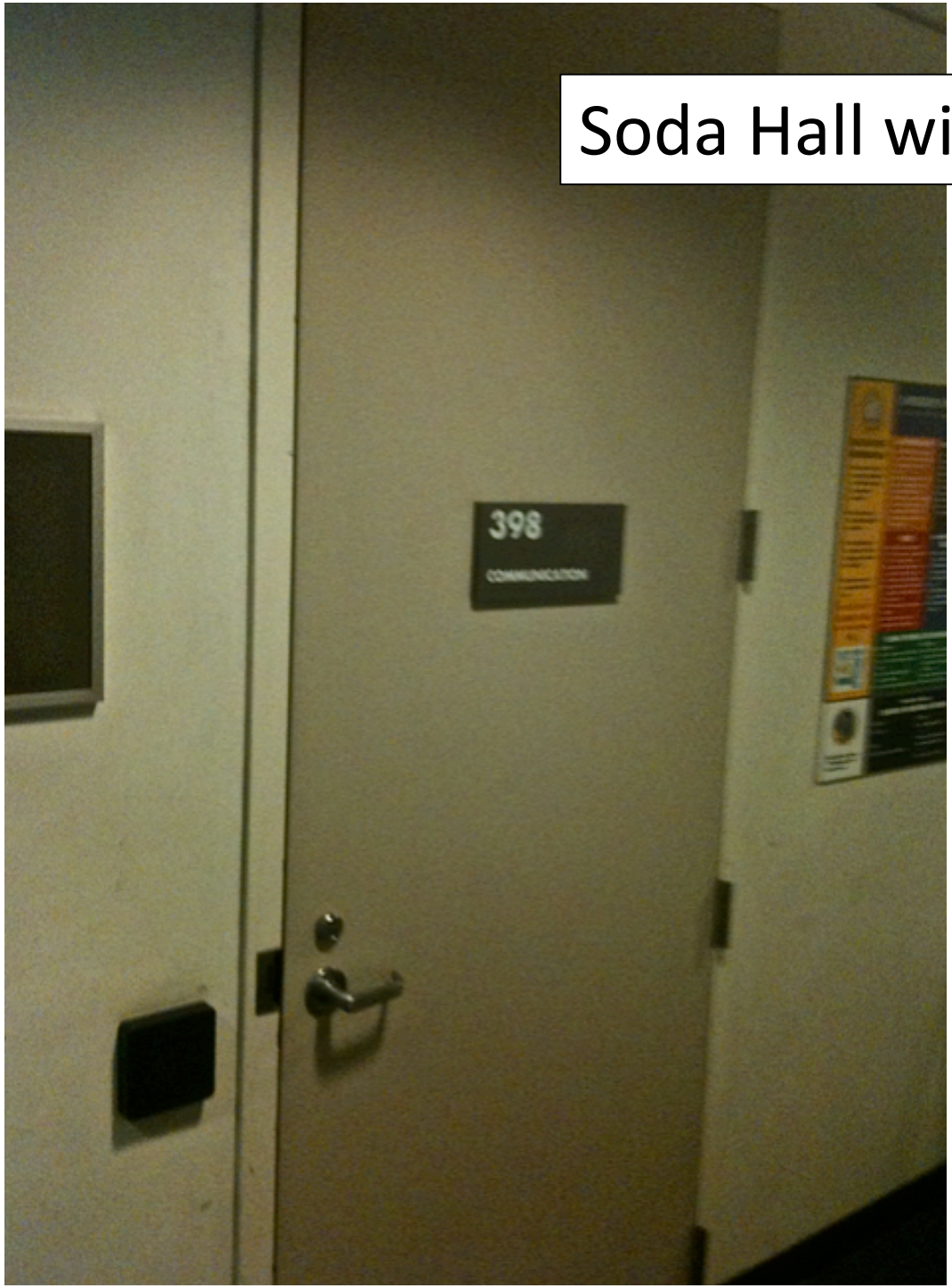
TXTL-60

“Security is economics.”

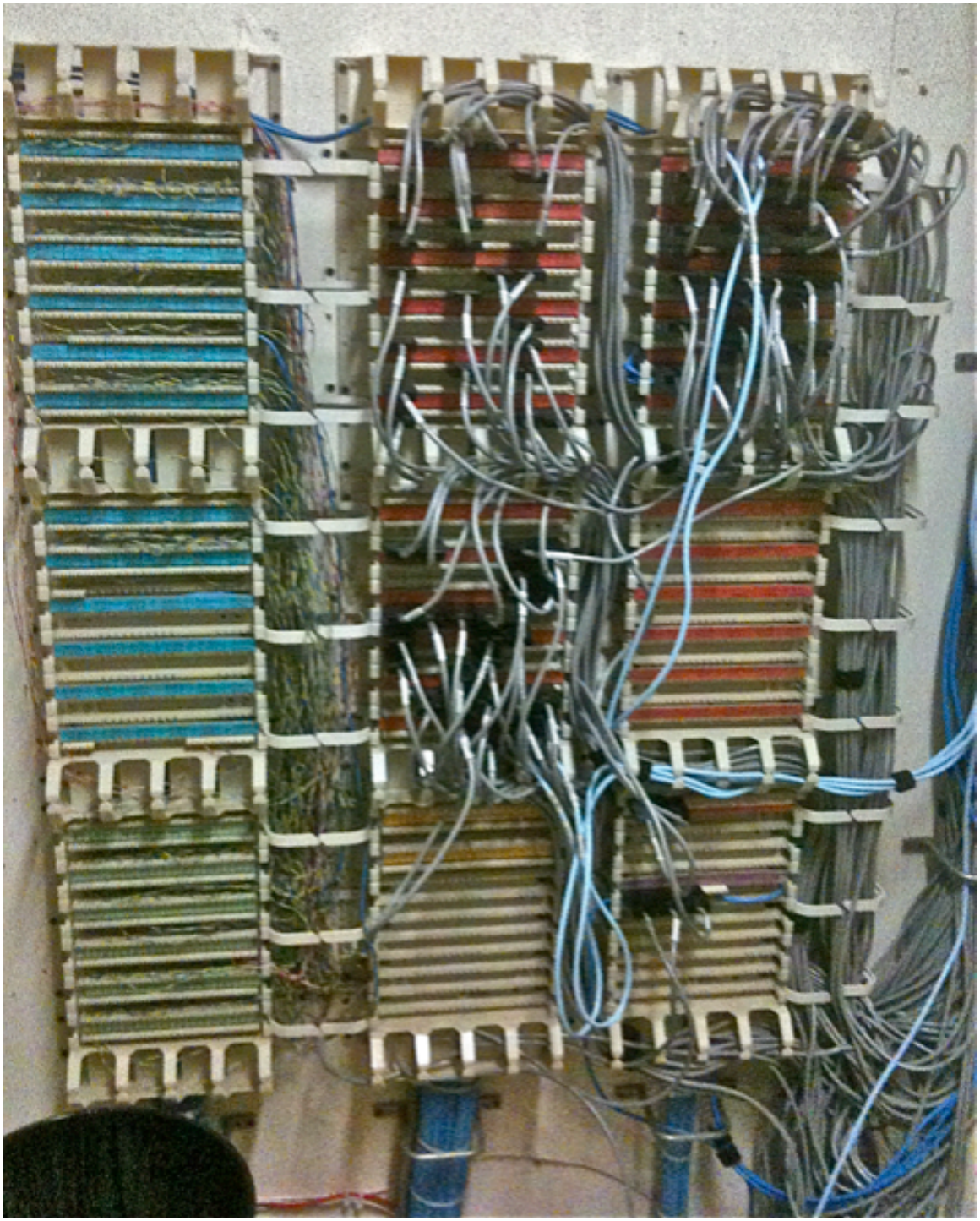
This program can delete any file you can.


Spider
Game
Deal!
Help
This program can delete any file you can.
Score: 461
Moves: 39

“Least privilege.”



Soda Hall wiring closet




398
COMMUNICATION

“Use fail-safe defaults.”

"inbound marketing" Search Advanced Search Preferences

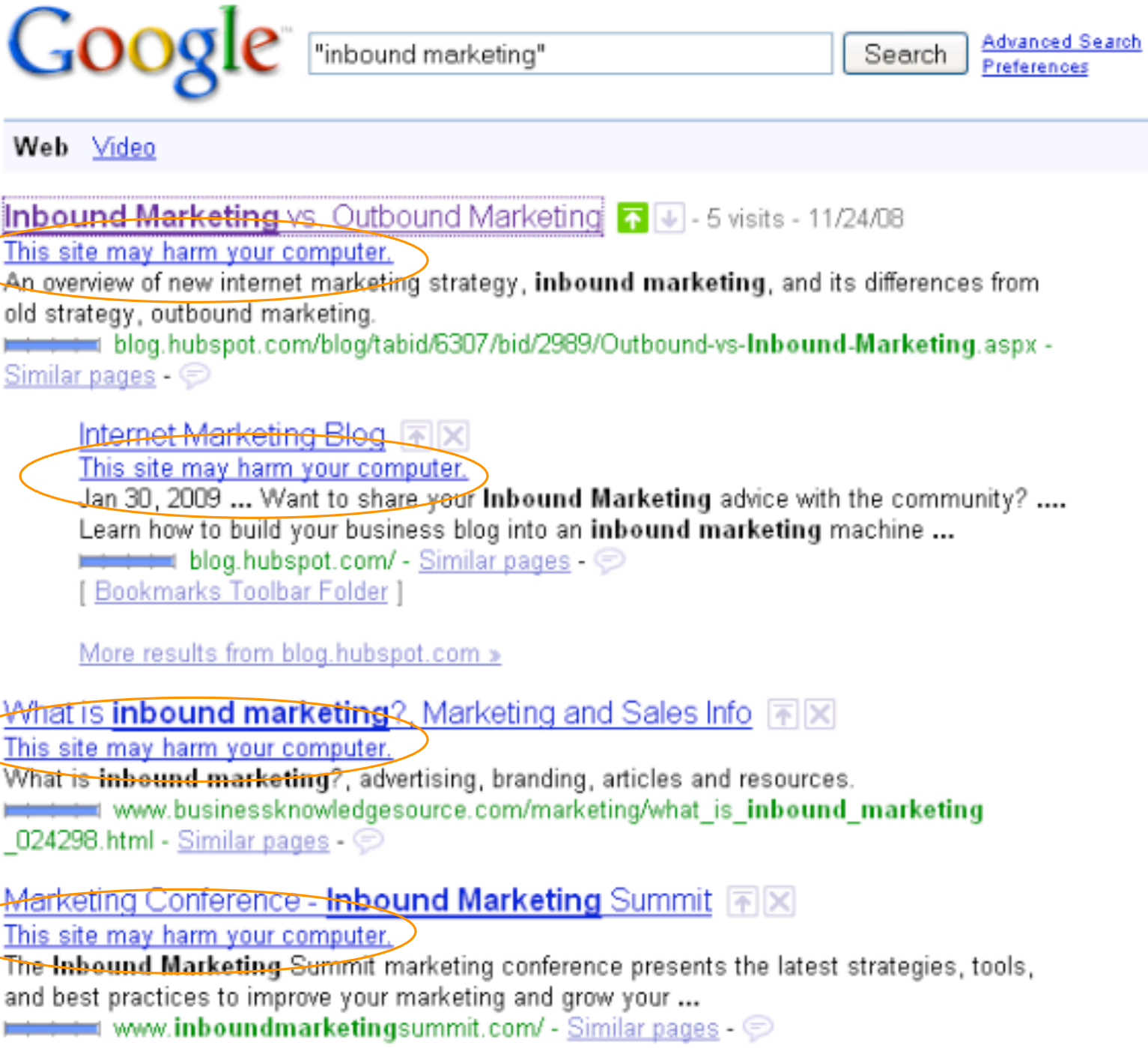
Web Video

Inbound Marketing vs. Outbound Marketing - 5 visits - 11/24/08

This site may harm your computer.

An overview of new internet marketing strategy, **inbound marketing**, and its differences from old strategy, outbound marketing.

blog.hubspot.com/blog/tabid/6307/bid/2989/Outbound-vs-**Inbound-Marketing**.aspx - Similar pages -


Google
"inbound marketing"
Search
Advanced Search
Preferences
Web
Video
Inbound Marketing vs. Outbound Marketing - 5 visits - 11/24/08
This site may harm your computer.
An overview of new internet marketing strategy, inbound marketing, and its differences from old strategy, outbound marketing.
blog.hubspot.com/blog/tabid/6307/bid/2989/Outbound-vs-Inbound-Marketing.aspx - Similar pages -
Internet Marketing Blog
This site may harm your computer.
Jan 30, 2009 ... Want to share your Inbound Marketing advice with the community? .... Learn how to build your business blog into an inbound marketing machine ...
blog.hubspot.com/ - Similar pages -
[Bookmarks Toolbar Folder]
More results from blog.hubspot.com »
What is inbound marketing?, Marketing and Sales Info
This site may harm your computer.
What is inbound marketing?, advertising, branding, articles and resources.
www.businessknowledgesource.com/marketing/what_is_inbound_marketing_024298.html - Similar pages -
Marketing Conference - Inbound Marketing Summit
This site may harm your computer.
The Inbound Marketing Summit marketing conference presents the latest strategies, tools, and best practices to improve your marketing and grow your ...
www.inboundmarketingsummit.com/ - Similar pages -

The Official Google Blog

Insights from Googlers into our products, technology, and the Google culture.

"This site may harm your computer" on every search result?!?!

1/31/2009 09:02:00 AM

If you did a Google search between 6:30 a.m. PST and 7:25 a.m. PST this morning, you likely saw that the message "This site may harm your computer" accompanied each and every search result. This was clearly an error, and we are very sorry for the inconvenience caused to our users.

What happened? Very simply, human error. Google flags search results with the message "This site may harm your computer" if the site is known to install malicious software in the background or otherwise surreptitiously. We do this to protect our users against visiting sites that could harm their computers. We maintain a list of such sites through both manual and automated methods. We work with a non-profit called StopBadware.org to come up with criteria for maintaining this list, and to provide simple processes for webmasters to remove their site from the list.

We periodically update that list and released one such update to the site this morning. Unfortunately (and here's the human error), the URL of '/' was mistakenly checked in as a value to the file and '/' expands to all URLs. Fortunately, our on-call site reliability team found the problem quickly and reverted the file. Since we push these updates in a staggered and rolling fashion, the errors began appearing between 6:27 a.m. and 6:40 a.m. and began disappearing between 7:10 and 7:25 a.m., so the duration of the problem for any particular user was approximately 40 minutes.