Datasäkerhetsmetoder föreläsning 5

Examination genom säkerhetsutvärdering
Mjukvarusäkerhet, lagar

Jan-Åke Larsson

LINKÖPING UNIVERSITY

Examination genom säkerhetsutvärdering

- Examination sker genom en projektuppgift
- Genomförs ensam eller i par om två personer
- Uppgiften består av en beskrivning av en enkel fiktiv situation
- ... och man förväntas analysera situationens datasäkerhet
- En förslagslista på ämnen finns på Lisam

Innehåll säkerhetsutvärdering

- CIA-analys
- Hotanalys
- Riskanalys
- Bristidentifiering
- Åtgärdslista med prioritering

Storlek och bedömning säkerhetsutvärdering

- Normalt sidantal inlämning är 8-10 sidor
- Det som bedöms är om innehållet är tillräckligt.
- Lämnas in i Lisam i något standardformat
- Vi ser allvarligt på frågan om plagiering och använder Urkund som kontroll

Betyg säkerhetsutvärdering

- Uppgiften bedöms med graderade betyg, U, 3, 4 eller 5.
- För betyget 3 krävs att rapporten är läsbar, och innehåller alla nödvändiga delar.
- För betyget 4 krävs att rapporten är bra nog att stöd av en äldre kollega räcker för att man ska kunna använda rapporten på riktigt.
- För betyget 5 krävs att rapporten är bra nog att det enda som behövs är mer arbetstid av er för att man ska kunna använda rapporten på riktigt.
- Vid betyget U så finns en chans att komplettera till betyg 3. Övriga betyg kan ej höjas genom komplettering.

Viktiga datum

- Första utkast senast 12 december
- Återkopping fö 10
- Slutversionen lämnas in senast 11 januari 2018
- Vi examinerar också i påskperioden och i augusti

Software security

- Security and reliability are both about unexpected problems
- Reliability is about purely accidental failures
- The probability that a failure will happen follows a certain distribution
- It does not matter how many bugs there are, it matters how often they are triggered
- Testing is against expected usage

- Attackers do the unexpected, and effectively chooses the distribution
- You'd want your code bug-free
- Experience shows that the number of bugs decrease exponentially

Malware taxonomy

- *Malware* is any software with a malicious purpose
- *Computer viruses* are self-replicating code pieces that *infects* other legitimate programs and files
- *Worms* are self-replicating code pieces that spread on their own
- *Trojan horses* are programs with a legitimate purpose but also with hidden malicious functions

Hackers

- *Hackers* initially referred to people with intimate knowledge about programming and computer systems
- *Crackers* was the term for people that performed attacks on computer systems
- Nowadays, “hacker” has a negative connotation
- Note, though, the distinction between *white hat* and *black hat* hackers
- These days, purely criminal organizations are more and more active

- Don’t try this at home

Dangers in change, and in abstraction

- Change is a big problem
- Even if (you think) you understand the implications of a change, it is easy to get it wrong

- Abstraction is very useful to understand complex systems
- However, security implications are often so detail-dependent that hiding the details can be a big problem
- Sometimes, the abstraction does not correspond to the actual implementation
- It is very important to be clear about the threats; these often drop from view in abstract models

Abstraction threats: Characters

- You want to give access only to `/A/B/C`, and your application appends the `input` into `/A/B/C/input`
- Attacker enters `../../../etc/passwd`
- Input validation should be used

- UTF-8 specifies %c0%af='/'
- An old Microsoft IIS accepted this, so that `[IP]/scripts/..%c0%af../winnt/system32/` decoded to `C:\winnt\system32`

Abstraction threats: Characters

- UTF-8 specifies %c0%af='/'
- An old Microsoft IIS accepted this, so that `[IP]/scripts/..%c0%af../winnt/system32/` decoded to `C:\winnt\system32`

- There is a further twist, since the decoding is to binary and then a further string decoding
- The string `[IP]/scripts/..%25%32%66../winnt/system32/` decodes to `[IP]/scripts/..%2f../winnt/system32/` which decodes to `[IP]/scripts/../../winnt/system32/`

- Decoding UTF-8 is translation between levels of abstraction

Abstraction threats: Integers

- 8-bit integers: $255 + 1 = 0$
- 8-bit signed integers: $127 + 1 = -128$, $-128/-1 = -128$
- Type confusion: 255 (unsigned) $= -1$ (signed)
- The comparison

  `if (size < sizeof(buf))`

  might be true: if `size` is signed and you assign a large value, it might turn negative

- UNIX once contained programs that first checked that the UID is not zero(=root), and then truncated the UID to an `unsigned short`

Abstraction threats: Integers

- Basic problem: computer integers are not mathematical integers
- $b \geq 0 \not\Rightarrow a + b \geq a$
- Use unsigned integers, watch out for integer overflow
- Turn on compiler warnings for signed-unsigned comparison

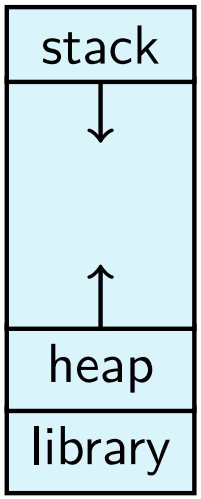
Canonicalization

- Filenames have several different but equivalent representations
- Dotless IP have 32 bits: $a.b.c.d = 2^{24}a + 2^{16}b + 2^8c + d$
- Symbolic (soft) links give more equivalent representations
- Some systems have case-insensitive filenames (for example, consider old Apache on HFS+)

- Perform access-control decisions in one unique canonical representation

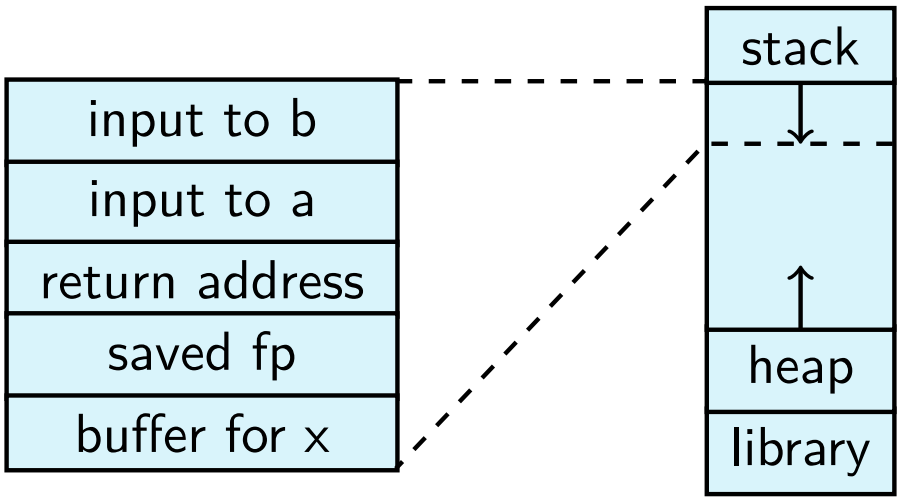
Memory management

- Buffer overruns
  - Stack overruns
  - Heap overruns
- Double-free vulnerabilities



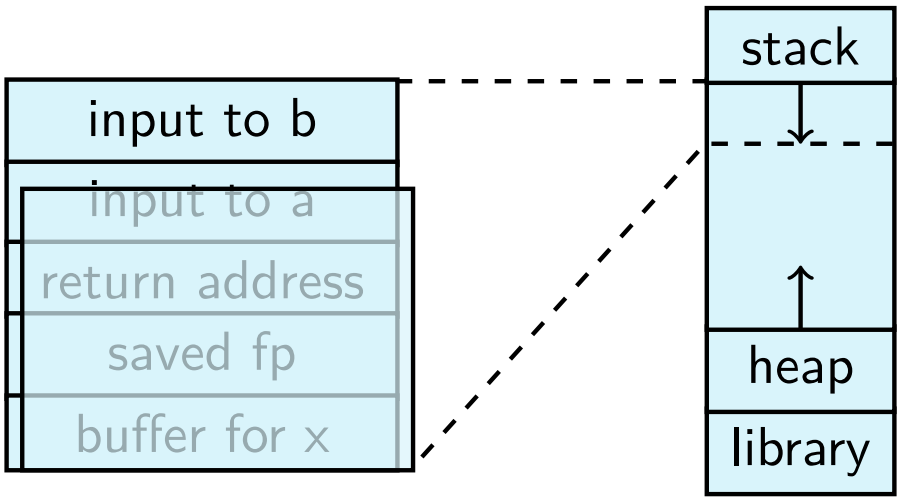
Stack overruns

```
void myfunction(int a, int b)
{
  char x[20];
  ...
}
```



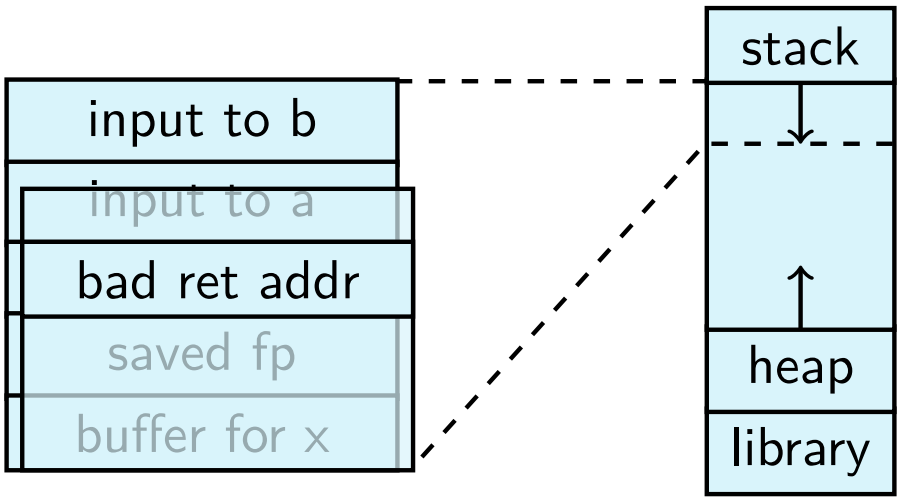
Stack overruns

```
void myfunction(int a, int b)
{
  char x[20];
  ...
}
```



Stack overruns

```
void myfunction(int a, int b)
{
  char x[20];
  ...
}
```



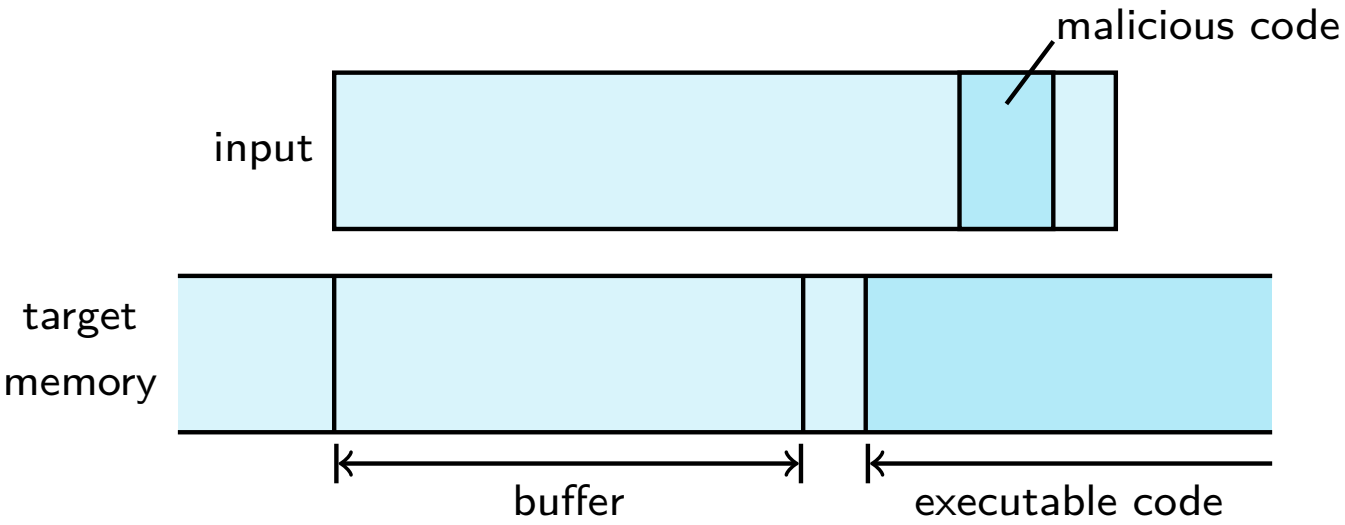
The 1980s — the era of personal computers

- The "Morris worm" of 1988 infected 5-10% of all machines connected to the internet
- Used a buffer overrun in the *fingerd* daemon of VAXes running BSD Unix
- Perpetrator sentenced to $10000 fine and 400 hours community service

```
push1 $68732f          push '/sh, <NUL>'
push1 $6e69622f        push '/bin'
mov1  sp, r10          save stackp in r10 (string beginning)
push1 $0               push 0 (arg 3 to execve)
push1 $0               push 0 (arg 2 to execve)
push1 r10              push string beginning (arg 1 to execve)
push1 $3               push argc
mov1  sp, ap           set argv to stackp
chmk  $3b              perform 'execve' kernel call
```

Buffer overruns, and "smashing the stack"

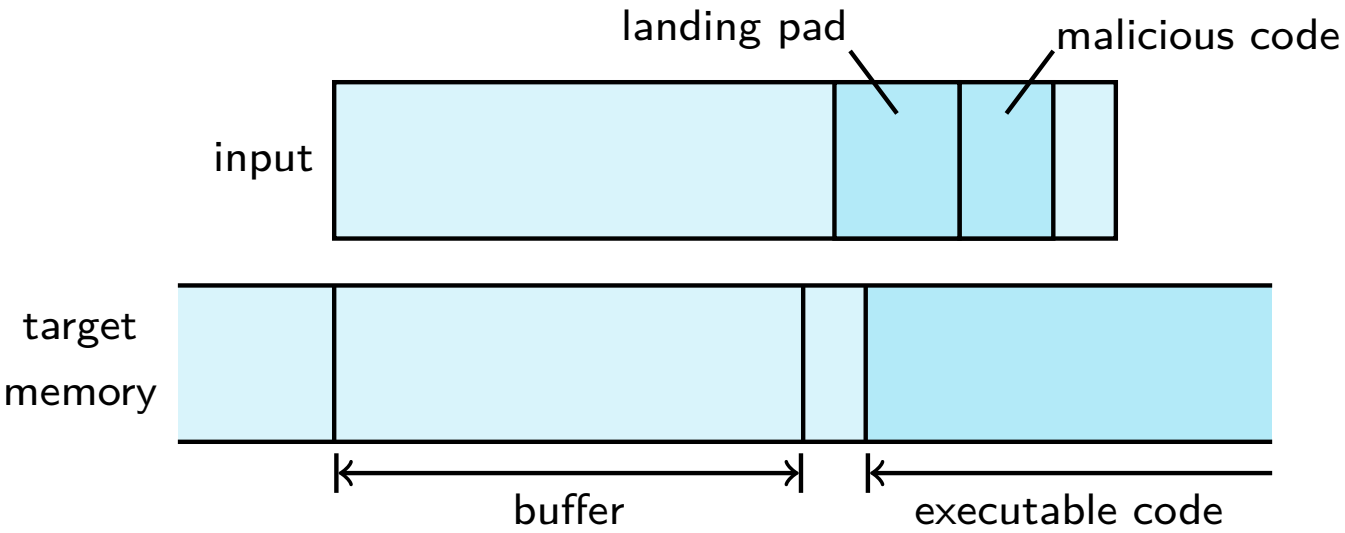
- Basic weakness: programmers are often careless about checking the size of arguments
- An attacker passes a long argument can find that some of it is treated as executable code rather than data



- About half of the CERT bulletins are (were 2008) of this kind, this is slowly decreasing

Buffer overruns, and "smashing the stack"

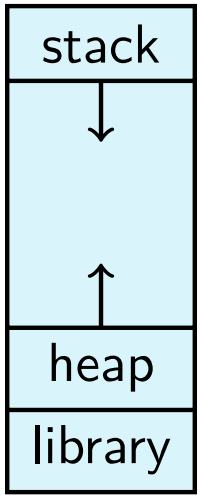
- Basic weakness: programmers are often careless about checking the size of arguments
- An attacker passes a long argument can find that some of it is treated as executable code rather than data



- About half of the CERT bulletins are (were 2008) of this kind, this is slowly decreasing

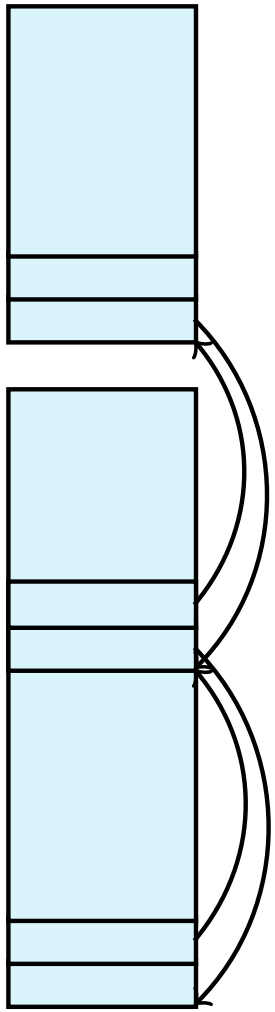
Heap overruns

- These are harder to perform since it is more difficult to predict where the buffer is in relation to the target
- Target is usually pointers
  - Pointers to open files
  - Pointers to functions (requires executable heap)
- Effect can be crash rather than break, but with enough attempts (large vulnerable user base), the attack will eventually result in a break



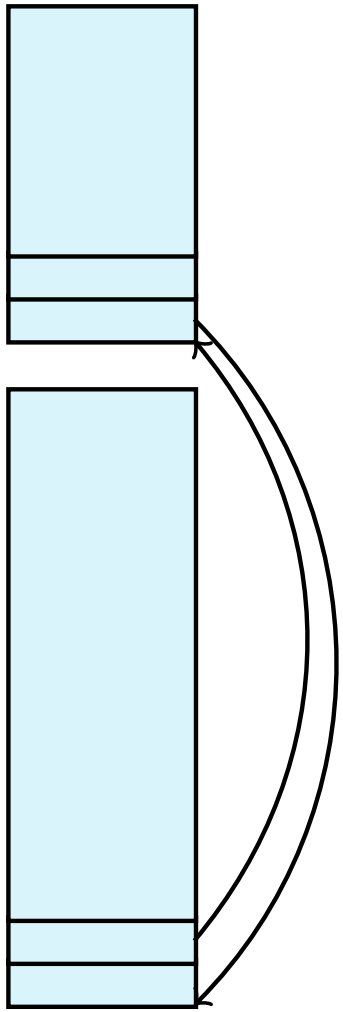
Double-free vulnerabilities

- In a double-free vulnerability, the OS itself is lured into writing into the target memory location
- If memory is free'd but the pointer not zeroed, it could be free'd again
- The function `malloc` allocates a chunk of memory
- The function `free` gives it back to the system
- Free memory is kept in a double-linked list
- There is a mechanism to join chunks back together, and this is the weakness



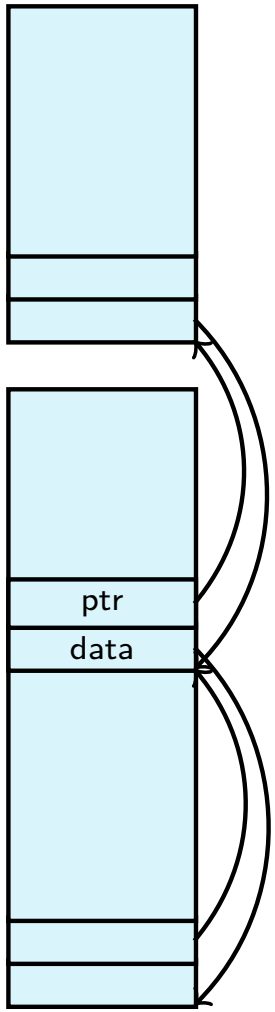
Double-free vulnerabilities

- In a double-free vulnerability, the OS itself is lured into writing into the target memory location
- If memory is free'd but the pointer not zeroed, it could be free'd again
- The function `malloc` allocates a chunk of memory
- The function `free` gives it back to the system
- Free memory is kept in a double-linked list
- There is a mechanism to join chunks back together, and this is the weakness



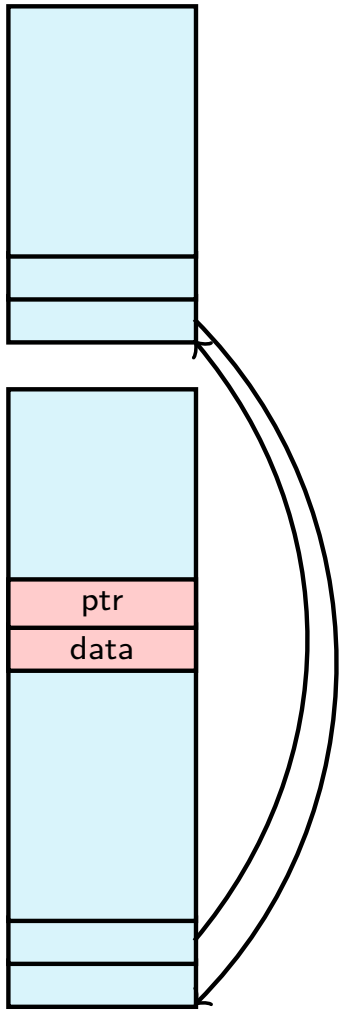
Double-free vulnerabilities

- In a double-free vulnerability, the OS itself is lured into writing into the target memory location
- If memory is free'd but the pointer not zeroed, it could be free'd again
- The function `malloc` allocates a chunk of memory
- The function `free` gives it back to the system
- Free memory is kept in a double-linked list
- There is a mechanism to join chunks back together, and this is the weakness



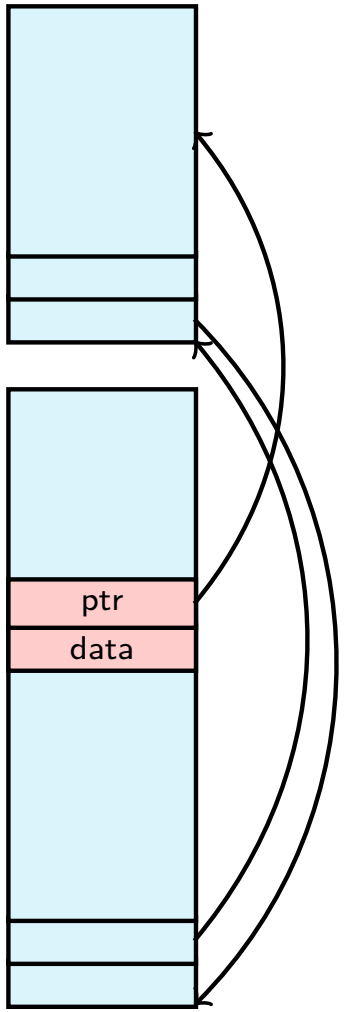
Double-free vulnerabilities

- In a double-free vulnerability, the OS itself is lured into writing into the target memory location
- If memory is free'd but the pointer not zeroed, it could be free'd again
- The function `malloc` allocates a chunk of memory
- The function `free` gives it back to the system
- Free memory is kept in a double-linked list
- There is a mechanism to join chunks back together, and this is the weakness



Double-free vulnerabilities

- In a double-free vulnerability, the OS itself is lured into writing into the target memory location
- If memory is free'd but the pointer not zeroed, it could be free'd again
- The function `malloc` allocates a chunk of memory
- The function `free` gives it back to the system
- Free memory is kept in a double-linked list
- There is a mechanism to join chunks back together, and this is the weakness



Distinguishing data and code: scripting

- The importance of sanitizing input cannot be underlined too much

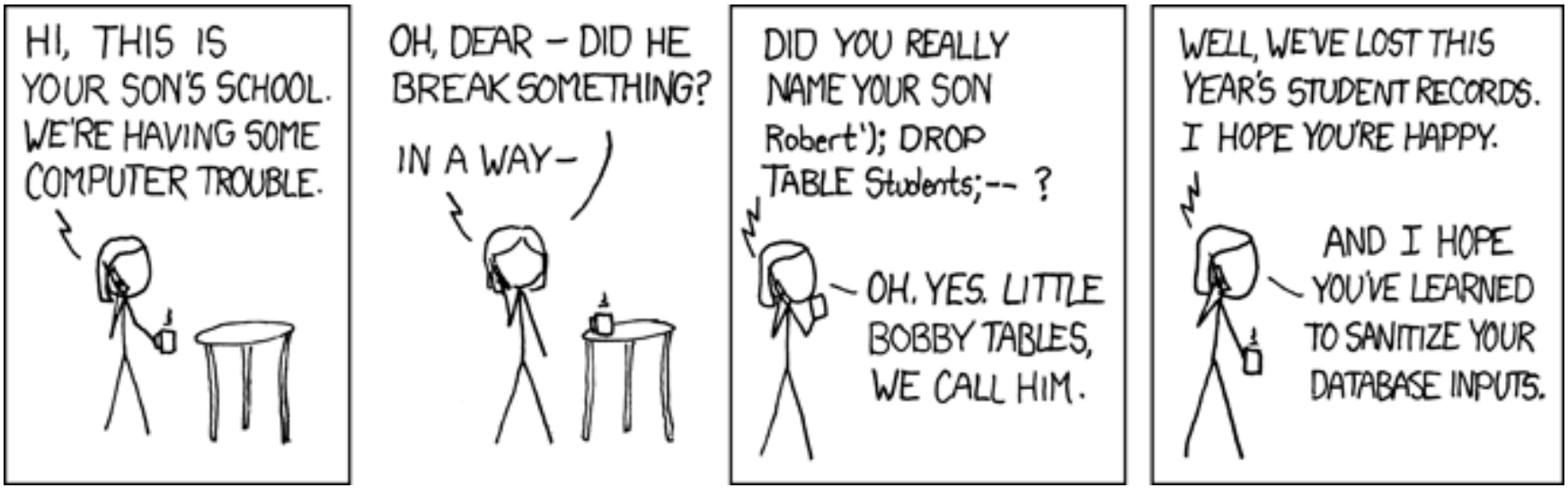
```
#!  /bin/bash
cat $1 | mail $2
```

- Call this with the following

```
foo thefile 'nobody@home | rm -rf /'
```

- Or rather, don't!

SQL injection

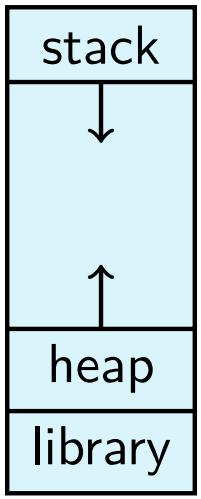


Race conditions

- Strange things can happen when multiple processes or threads access the same data
- In CTSS (a time-sharing OS from the 60s), a user found that the "message of the day" contained the password file
  - CTSS was designed for low memory, and had a tempfile for the editor named "SCRATCH" in the home directory
  - This was no problem since the home directory was writable only by the owner
  - Later, the *system* user was allowed to be used by several people
  - One edits "message of the day", another edits the password file ...
- Another example of TOCTTOU (if in different clothes)
- Can be prevented by using *file locking*
- You can still find this in modern systems as unsafe tempfile handling

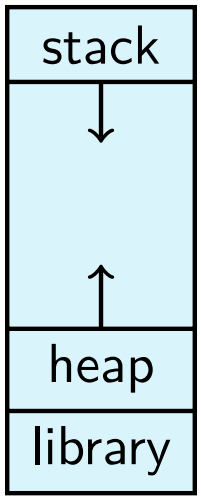
Prevention: Hardware

- An example is Intel's Itanium that has a separate register for the return address
- A more extrems solution is to put the return address in a separate Secure Return Address Stack
- This kind of hardware protection does not need recompilation
- But more extensive changes (to processor instructions, for example) may need changes in multi-threaded programs



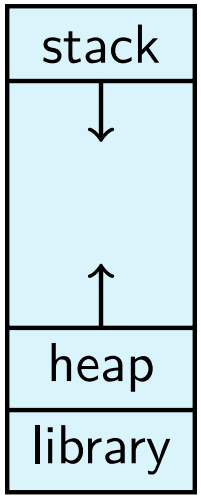
Prevention: Modus operandi

- A non-executable stack stops certain attacks
- Software that requires executable stack will stop working
- It is to the attacker's advantage if memory usage is predictable
- Address space layout randomization can prevent this
- BSD has used this to prevent argv[] attacks
- Windows uses this in system libraries as a defence against return-to-libc attacks



Prevention: Safer functions

- C is infamous for its string handling functions, say `strcpy`, `sprintf`, or `getc`
- For `strcpy`, the result is undefined if strings are not null-terminated
- There is no check if the destination buffer is long enough
- The function `strncpy` is better, since there is a count argument for the longest string length
- But watch out! this does not put a null at the end of the string
- Also watch out for integer overflows
- Perhaps use `bstrlib`?



Prevention: Filtering

- Whitelisting is the safer option
- Blacklisting is more difficult to get right
  - You must know about all dangerous inputs
  - . . . in all encodings (UTF-7 has been used in XSS attacks)
  - "Helpful" system components may trip you, say converting a foreign character into $<$ or '
- Filtering is difficult and complex
- (In)famous example: "Medireview".
  - Yahoo! Mail wanted to stop a JavaScript bug in their webmail system back in 2001
  - Filtering was done by replacing "eval" with "review"
  - However, this resulted in English words such as "medieval" being replaced by "medireview".

Prevention: Type safety

- There are programming environments (and compilers) that check unsafe usage of types
  - *Dynamic* type checking checks at runtime, and slows the program down
  - *Static* type checking does the checking in advance, at compile time; this requires more complicated checking, but does not decrease performance at runtime
- What is often ensured is memory integrity
- We would want to get execution integrity; may be difficult to specify exactly what this means

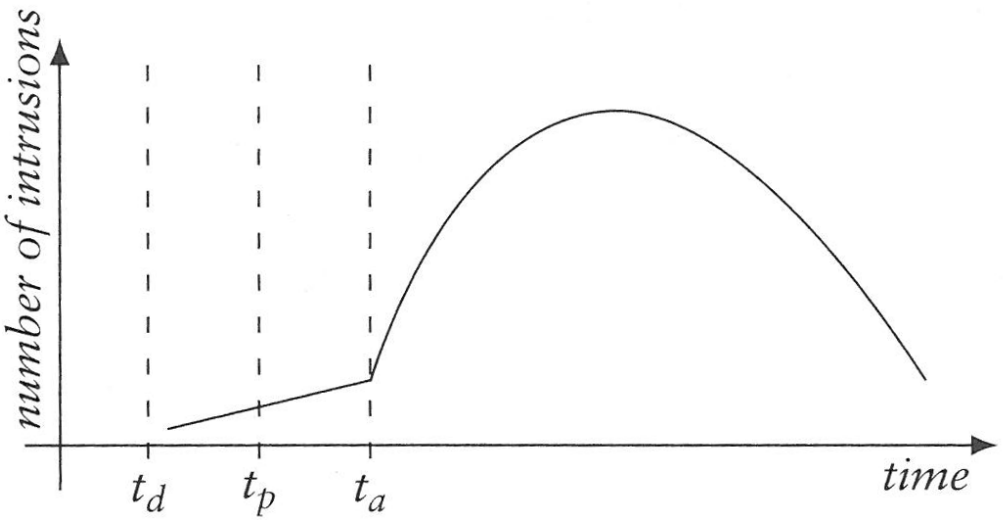
Detection

- *Canaries* are memory elements used to detect unwanted changes to memory at runtime
- *Code inspection* by hand is slow and error prone, but does help somewhat
- *Automated code inspection* uses an expert system with known weaknesses
- Security *testing* does not need the source code, but may use the specification of allowed inputs and expected outputs
  - Random inputs are not so useful, since the attacker should choose the distribution
  - Common attacks needs to be tested; there are several examples of this

Mitigation: Least privilege

- Be sparing with requiring priviliges to run the code
- Do not give users more rights than needed
- Drop rights immediately when possible
- Do not activate options you do not need

Reaction: keeping up to date



t_d . . . time of disclosure
t_p . . . time patch released
t_a . . . time attack script released

Security requirements from laws

- There isn't a "computer security law"
- But there is a collection of laws that have direct implications on computer security, and the technology we use for it
- Here follows some national examples related to common international principles

Laws regulating computer security typically concern

- The security of the nation (e.g., secrecy for defence data)
- Personal privacy (e.g., secrecy for health data etc.)
- Bookkeeping (data integrity requirements)
- Context dependent limits to what may be processed and how (Data acts concerning private personal data, laws regulating medical journals etc.)

Offentlighets- och sekretesslagen (public availability and secrecy)

- Some information must be publicly available
- Other information can (must) be kept secret
- The latter concerns mostly national security (defence, economy, . . .) and privacy of citizens
- Some of the secrecy requirements do apply to private companies and organisations

The principle of public access to information, Offentlighetsprincipen

- Originates in the press freedom law from 1766
- Regulates citizens' rights to scrutinise public authorities' acts and decisions
- Everyone has a right to read all public documents ("allmän handling"): every document that has been created within or sent to a public authority except explicitly secret ones
- Swedish public authorities must make it easy for everyone to read these

Offentlighets- och sekretesslagen (public availability and secrecy)

The law has seven sections

1. Table of contents, applicability, definitions
2. Regulates “allmän handling”, searching, registering, labeling, and distribution
3. General rules on secrecy like against who, how to make exceptions, and how to handle responsibility
4. Specifics on national security, national economic policy, public auditing (and also nature preservation)
5. Specifics on individual security, like documents on health, taxes, support, legal issues, ...
6. Specifics on government security
7. Relation to other laws, and secrecy imposed by those

Bookkeeping laws, Bokföringslagen

- Regulates bookkeeping
- The documentation must be kept in an enduring and reliable way (>7 years)
- Documents and reading devices must be kept within Sweden (some exceptions exist, mainly for EU contries)
- Corrections must be visible and accompanied with a note on when and by whom the correction was made
- Electronically received material (invoices etc) must be archived in the form they were received

Säkerhetsskyddslagen, säkerhetsskyddsförordningen

- Regulate security (mostly confidentiality) for the national defence and its suppliers
- Classification of data, persons, and protection, especially how to approve persons for access to classified data
- Säkerhetsskyddsförordningen refers to detailed technical instructions about relevant threats and precautions

EU directive on protection of personal data

- Directive 95/46/EC of 24 October 1995
- Protect personal data against accidental or unlawful destruction or accidental loss, alteration, unauthorised disclosure or access
- Applies for storage, processing, or transmission of data over a network
- In Sweden, in the form of Personuppgiftslagen

The Swedish Data Act Personuppgiftslagen, PUL

- Concerns data about individuals (not companies, real estate data and suchlike, even when data on those concern just one person)
- Data can be kept and treated for individuals that
  - are customers, members etc.
  - have given explicit consent
  - are subject to research
  - are registered due to a "public interest" (requirements from authorities or laws)

Limitations to processing, PUL

- Only data relevant for the register, and the goal must be explicit
- "Sensitive data" (health, religion, political views, ethnicity, . . .) can only be kept by appropriate organisations (hospitals, churches, . . .)
- There must be procedures to ensure correctness of data
- Transfer of data to other countries is forbidden unless to a EU country, a country explicitly approved by the government or if the transfer benefits the subject

Security and “PUL”

- Security precautions are regulated in §31
- Requires “appropriate” technical and organisational precautions aimed at protecting data, as in
  a) the technical possibilities available,
  b) what it would cost to implement the measures,
  c) the special risks that exist with processing of personal data, and
  d) how sensitive the processed personal data really is.

Repetition: Cookies and privacy

- Cookies don't just store SIDs, they can be used to store all sorts of things
- But cookies are only sent to the matching domain?
- And cookies are stored at the user? No problem then!

- Not so fast, accessibility is really the problem, *and cookies are sent to the matching domain*
- This is a distributed database, and there are law requirements for such a thing

The Cookie Law

- EU directive 2009/136/EC, really on consumer rights in electronic communication
- Implemented in Sweden in 2011 (“kaklagen”)
- Cookies used to be opt-out, are now opt-in
- . . . except cookies that are essential for the service
- The current recommendation is: audit your cookies, be transparent, and explain the choices

Electronic “documents”

- A “document” should have a provable origin, and the integrity should be ensured
- In the digital world, this requires digital signatures (do not confuse this with digitised handwritten signatures)
- EU directive 99/93/EC: digital signatures should be legally valid in the member states

Electronic “documents”

- In Sweden formalized in “Lag (2000:832) om kvalificerade elektroniska signaturer”
  - A digital signature should be connected to one signer,
  - should make identification possible,
  - can be created with tools that only the signee holds, and
  - is associated with electronic data in such a way that integrity can be checked

Laws regulating computer security typically concern

- The security of the nation (e.g., secrecy for defence data)
- Personal privacy (e.g., secrecy for health data etc.)
- Bookkeeping (data integrity requirements)
- Context dependent limits to what may be processed and how (Data acts concerning private personal data, laws regulating medical journals etc.)

Examination genom säkerhetsutvärdering

- Biljettsystem
- Butikssystem
- Distansarbete
- Egen erfarenhet
- Enmansfirman
- Forskargrupp
- Hasardspel
- Hemstyrsystem
- Hemtjänst
- Kemiindustri
- Lantbruk
- Lokalbank
- Morbror August
- Offertunderlag
- Skogshemmet
- Vaktbolag
- Veterinärklinik