Package manager security

Simon Hollingshead

22nd January 2014

Plan for the talk

1. Package manager 101
    - The package
    - The client
    - The repository
2. Communicating with a repository
    - Building up from first principles
    - Case study: Is that really how they do it?
3. Breaking package managers
    - Why our simple package manager is exploitable
    - How distributions keep their packages secure
    - How to bypass current package manager security
4. My Part II project
    - What ‘GPM’ is

A package is just a collection of files

```
gimp.deb
+- control.tar.gz
|  +- control
|  +- md5sums
|  +- postinst
|  +- postrm
+- data.tar.xz
|  +- usr
|     +- bin
|     |  +- gimp
|     +- lib
|     |  +- gimp
|     +- share
|        +- applications
|        +- man
+- debian-binary
```

- Generally just archives full of binaries and libraries
- Structure of archive is just a directory tree
- Installation is a mass-copy and optional install script
- Even APKs, iOS apps, and Windows store apps are ‘packages’

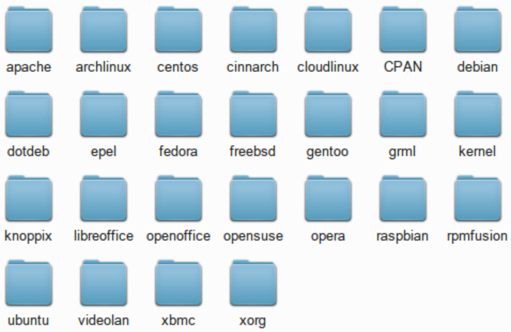
The client is a glorified file downloader

```
$ yum install gimp
Installing:  gimp
Installing for dependencies:  gtk3, libpng
```

A package manager is a utility that will:

- download software
- identify and fetch third-party dependencies
- detect updates
- use mirrors to spread load geographically
- use a library to perform the actual installation

The server does absolutely no computation

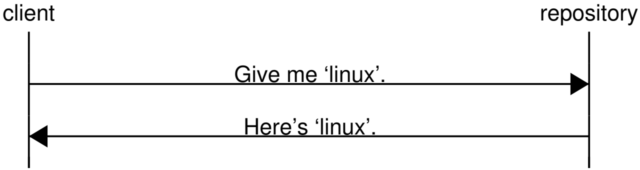


- A repository is just an HTTP or FTP server
- Multiple sites can be mirrored on one server
- One `rsync` command pulls all changes
- The easier it is, the more people will want to help out

1. Package manager 101
   - The package
   - The client
   - The repository
2. Communicating with a repository
   - Building up from first principles
   - Case study: Is that really how they do it?
3. Breaking package managers
   - Why our simple package manager is exploitable
   - How distributions keep their packages secure
   - How to bypass current package manager security
4. My Part II project
   - What ‘GPM’ is

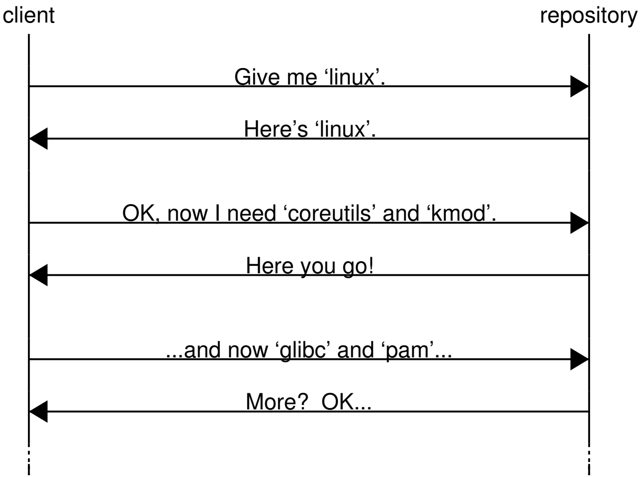
The most abstract view of the repository

You ask for something and you get it!



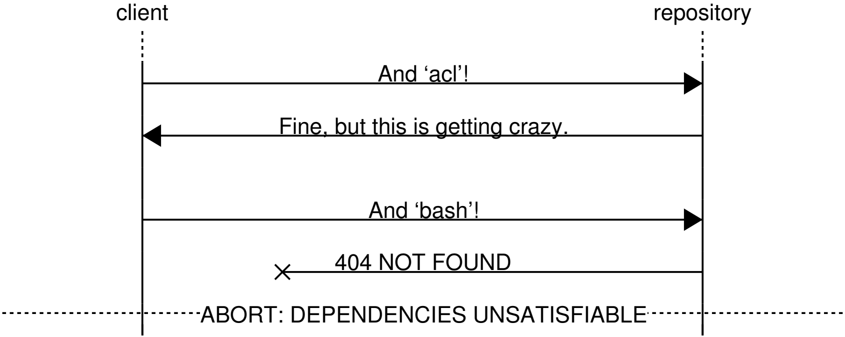
Package 'linux' has dependencies, though

Which you only learn about after the initial download...



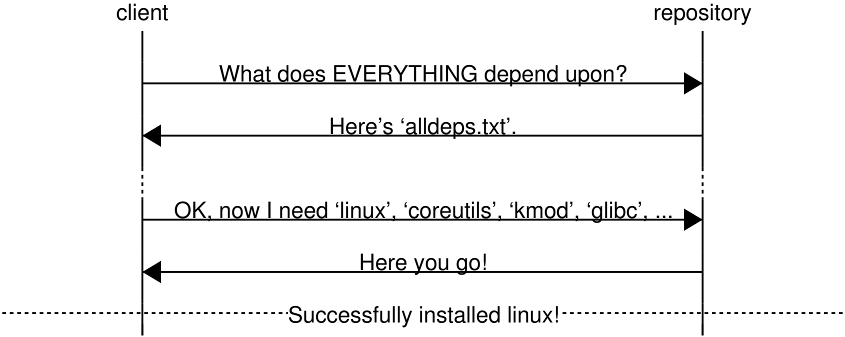
Worst case scenario – unsatisfiable dependencies

But only after downloading 100MB of other packages


client
repository
And ‘acl’!
Fine, but this is getting crazy.
And ‘bash’!
404 NOT FOUND
ABORT: DEPENDENCIES UNSATISFIABLE

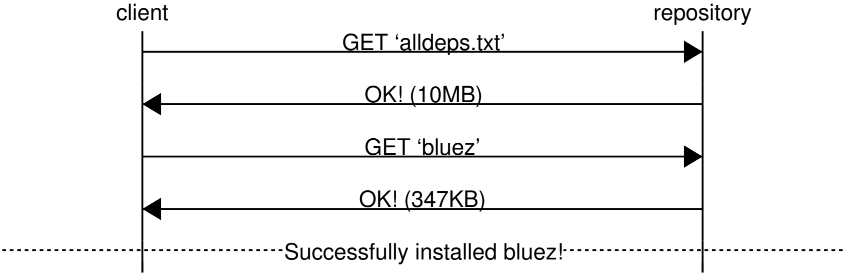
The best fix is to work out dependencies first

Don't start downloading files until we know it's OK


client
repository
What does EVERYTHING depend upon?
Here's 'alldeps.txt'.
OK, now I need 'linux', 'coreutils', 'kmod', 'glibc', ...
Here you go!
Successfully installed linux!

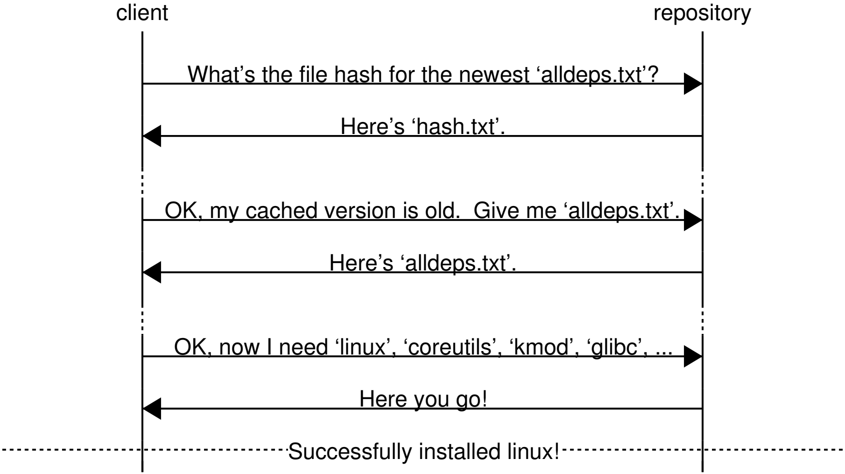
The dependency list is a very large file

So downloading it if it hasn't changed is wasteful



And now we have a good system!

Good enough to use in practice, even...



client

repository

What's the file hash for the newest 'alldeps.txt'?

Here's 'hash.txt'.

OK, my cached version is old. Give me 'alldeps.txt'.

Here's 'alldeps.txt'.

OK, now I need 'linux', 'coreutils', 'kmod', 'glibc', ...

Here you go!

Successfully installed linux!

Our 5 minute thought experiment had three stages

See if packages and metadata have changed

then

Fetch new metadata to allow dependency resolution

then

Download relevant packages

Let's try `apt-get update; apt-get install mtr-tiny`

See if packages and metadata have changed

```
$ sudo apt-get update
Get:1 ftp.uk.debian.org jessie InRelease [162 kB]
...
```

What's in 'ftp.uk.debian.org/debian/dists/jessie/InRelease'?

See if packages and metadata have changed

```
$ sudo apt-get update
Get:1 ftp.uk.debian.org jessie InRelease [162 kB]
...
```

What's in 'ftp.uk.debian.org/debian/dists/jessie/InRelease'?

```
d3a2e367c3171c6edf25f431250a38ac 12654818 main/binary-all/Packages
228afd3b80b42851f21268d3bfbd80f4 30638240 main/binary-amd64/Packages
2fbe443f6a3ec7b0d76c627aa167d9f2 29827127 main/binary-armel/Packages
e21318a2b6cc210cc52bfe132a95b277 29878415 main/binary-armhf/Packages
2a0d8a2af82d71f8852146dddd1c19ae 30649718 main/binary-i386/Packages
9e3b3932eddfaece831740d3db88efa9 28663190 main/binary-ia64/Packages
e077084a9167c94b535a672cfcaaa8e1 28805126 main/binary-kfreebsd-amd64/Packages
215d5197b3915424ee3f7572246a191a 28743453 main/binary-kfreebsd-i386/Packages
83e6ea1fc12497973d246e875e4d11e0 29536977 main/binary-mips/Packages
c48042aef6107ff41734ff181956d688 29650260 main/binary-mipsel/Packages
27cc98cad606c24a91a666493e130baa 30135352 main/binary-powerpc/Packages
27da2395d2265c90edd178b78170444f 28302234 main/binary-s390x/Packages
fe236d9db51fb375ba2621184005084e 29870665 main/binary-sparc/Packages
```

Fetch new metadata

```
...
Ign ftp.uk.debian.org jessie/main i386 Packages
```

and what's in '.../jessie/main/i386/Packages.gz'? (8MB)

Fetch new metadata

```
...
Ign ftp.uk.debian.org jessie/main i386 Packages
```

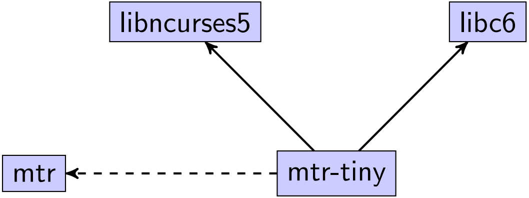
and what's in '.../jessie/main/i386/Packages.gz'? (8MB→31MB)

```
     1| Package: 0ad
   ...| ...
480342| Package: mtr-tiny
480343| Version: 0.85-2
480344| Installed-Size: 301
480345| Replaces: mtr
480346| Depends: libc6 (>= 2.15), libncurses (>= 5.7+20100313)
480347| Conflicts: mtr, suidmanager (<< 0.50)
480348| Filename: pool/main/m/mtr/mtr-tiny_0.85-2_i386.deb
480349| Size: 138610
480350| MD5sum: 293fb8b1b5af80ebf3b2f3833942f206
   ...| ...
760082|
```

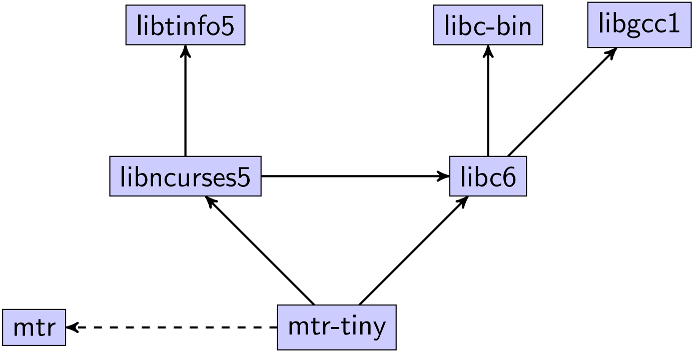
Dependency resolution

mtr-tiny

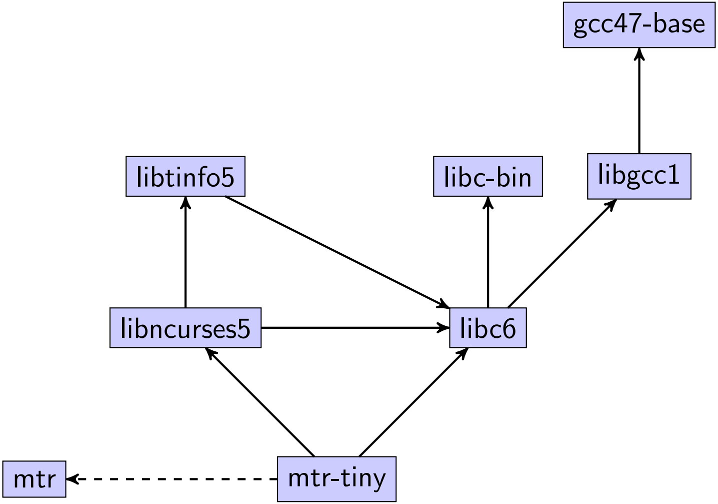
Dependency resolution



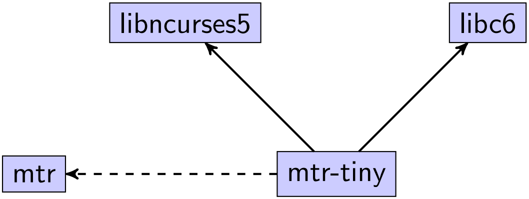
Dependency resolution



Dependency resolution



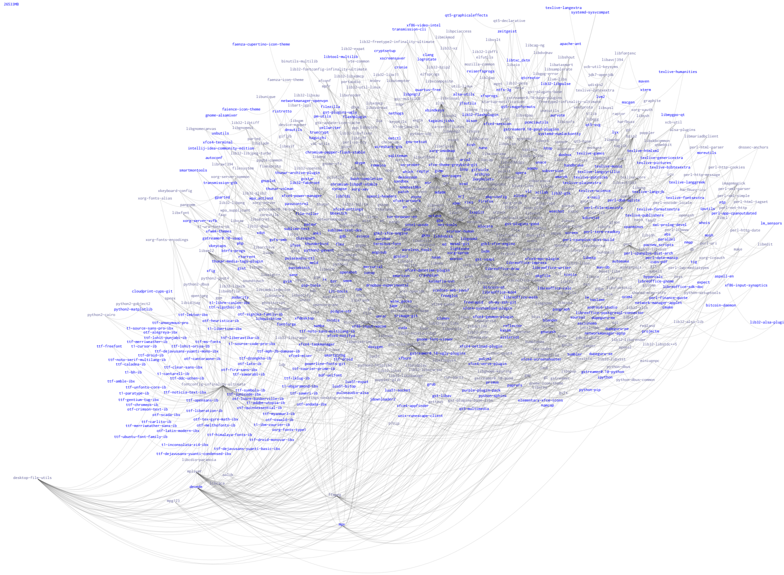
Dependency resolution



Download relevant packages

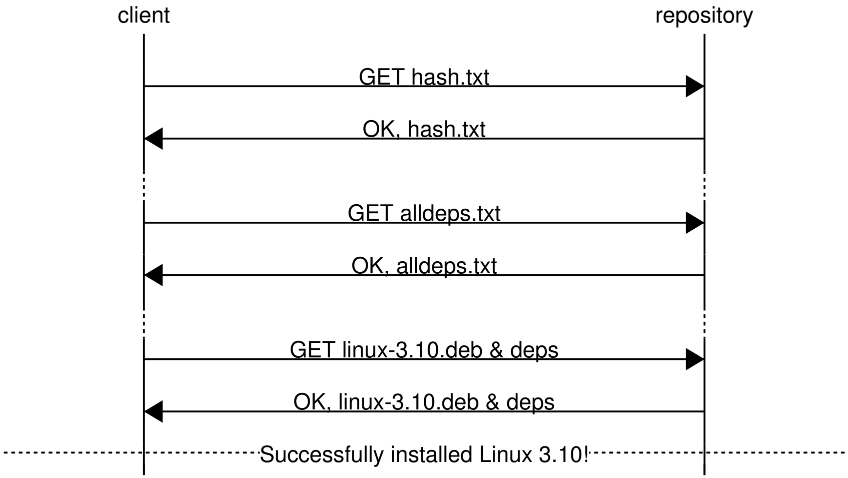
```
$ sudo apt-get install mtr-tiny
Reading package lists...  Done
Building dependency tree
Reading state information...  Done
The following packages will be REMOVED:
  mtr
The following NEW packages will be installed:
  mtr-tiny
0 upgraded, 1 newly installed, 1 to remove.
Need to get 139 kB of archives.
After this operation, 150 kB of additional disk space
will be used.
Get:1 ftp.uk.debian.org/debian jessie/main mtr-tiny
i386 0.85-2 [139 kB]
```

Dependencies quickly become complicated



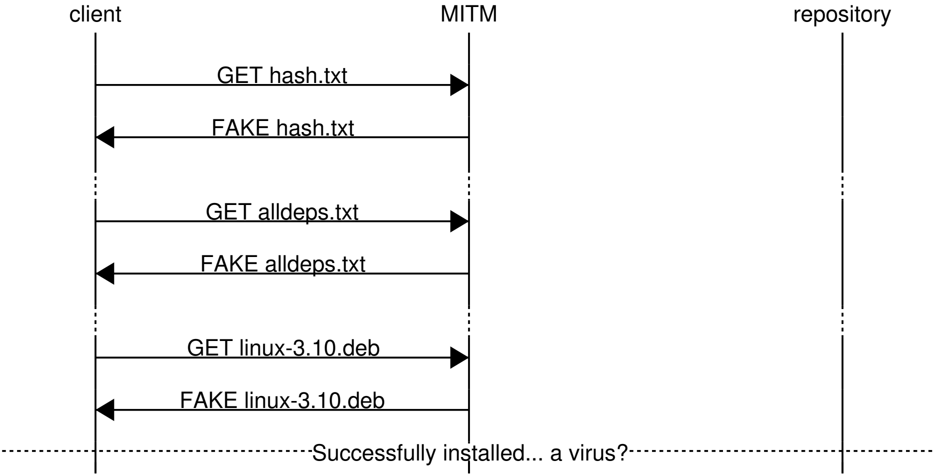
1. Package manager 101
   - The package
   - The client
   - The repository
2. Communicating with a repository
   - Building up from first principles
   - Case study: Is that really how they do it?
3. Breaking package managers
   - Why our simple package manager is exploitable
   - How distributions keep their packages secure
   - How to bypass current package manager security
4. My Part II project
   - What 'GPM' is

Earlier, we created this protocol


client
repository
GET hash.txt
OK, hash.txt
GET alldeps.txt
OK, alldeps.txt
GET linux-3.10.deb & deps
OK, linux-3.10.deb & deps
Successfully installed Linux 3.10!

Our idea is 100% INSECURE!

So were old versions of Arch Linux `pacman`/FreeBSD `ports`/Slackware `slaktool`



Two separate mechanisms fix man-in-the-middle attacks

SSL

Google Update, Firefox Updater, AIR apps, Sparkle

- Authenticates repository
- Generally costs money
- Harder to be a mirror
- Always decrypted in RAM
- 56 CAs + 7 governments

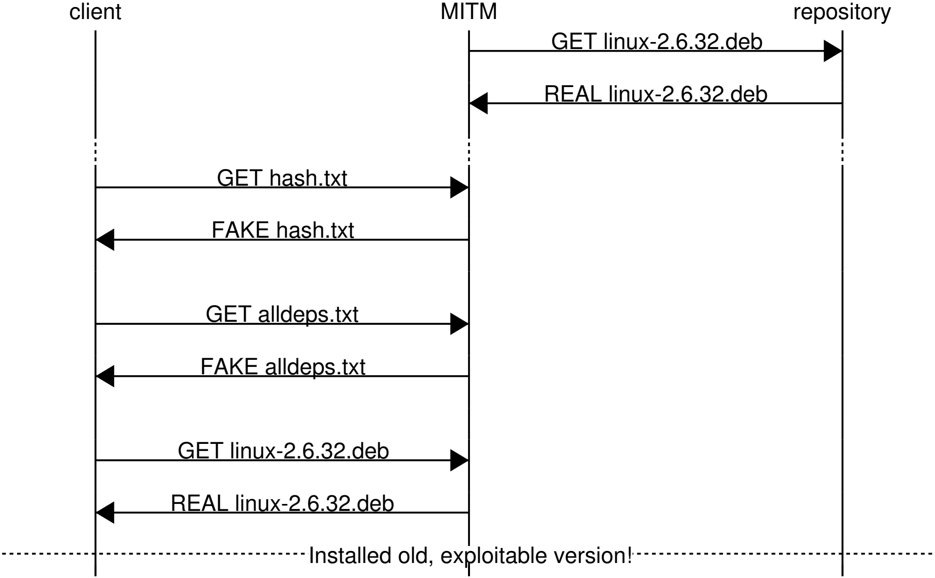
Package signatures

`apt`, `pacman`, `portage`, `urpmi`, `yum`, …

- Authenticates package
- Totally free
- Easy to mirror
- Decrypted only to sign
- Create your own ‘root’

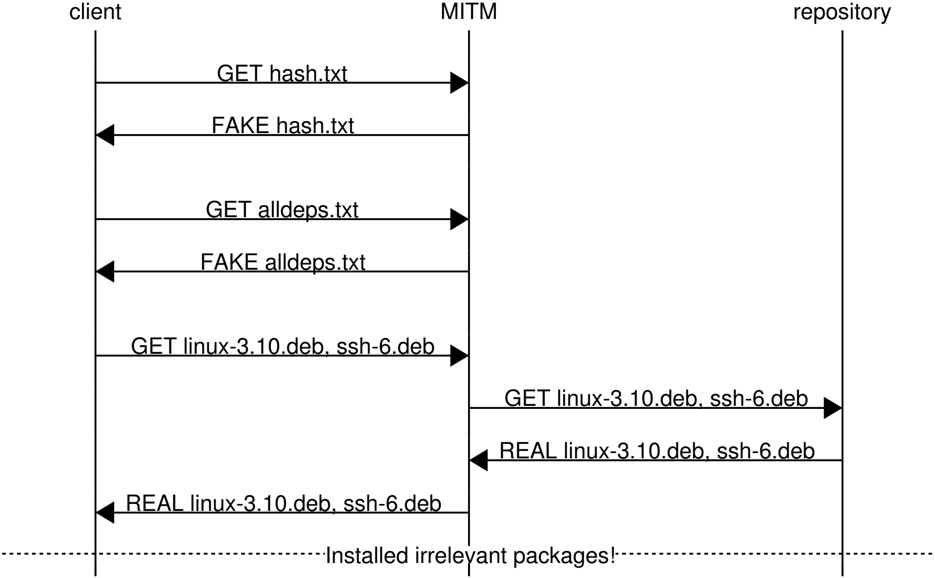
Signed packages 1 - Lie about newest version

Old CentOS `yum`/Mandriva `urpmi`... and CURRENT Arch Linux `pacman`!



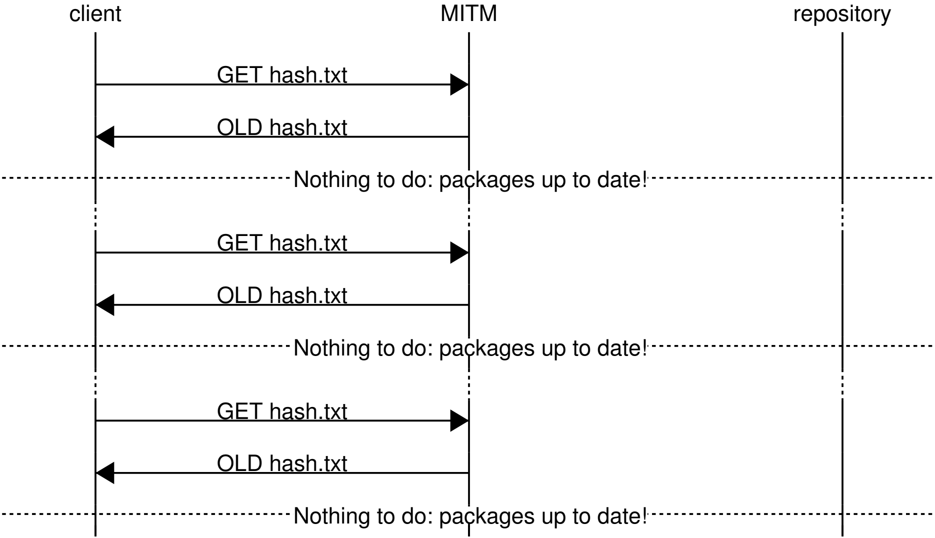
Signed packages 2 - Modify dependencies

Old CentOS `yum`/Mandriva `urpmi`... and CURRENT Arch Linux `pacman`!


client
MITM
repository
GET hash.txt
FAKE hash.txt
GET alldeps.txt
FAKE alldeps.txt
GET linux-3.10.deb, ssh-6.deb
GET linux-3.10.deb, ssh-6.deb
REAL linux-3.10.deb, ssh-6.deb
REAL linux-3.10.deb, ssh-6.deb
Installed irrelevant packages!

Signed packages & metadata - freeze updates

Current Gentoo portage and “Stork Secure Package Manager”


client
MITM
repository
GET hash.txt
OLD hash.txt
Nothing to do: packages up to date!
GET hash.txt
OLD hash.txt
Nothing to do: packages up to date!
GET hash.txt
OLD hash.txt
Nothing to do: packages up to date!

! WARNING !

Exploiting computers you don’t own is bad.

Exploiting your own VMs is fine, though.

...so here’s a demo.

The solution is clear – sign all three stages

This is perfect, if...

- Different keys sign each stage
- Keys expire at appropriate speeds
- Keys are revoked as a precaution
- Manual, insecure updates are restricted
- Keys are encrypted when not signing

1. Package manager 101
   - The package
   - The client
   - The repository

2. Communicating with a repository
   - Building up from first principles
   - Case study: Is that really how they do it?

3. Breaking package managers
   - Why our simple package manager is exploitable
   - How distributions keep their packages secure
   - How to bypass current package manager security

4. My Part II project
   - What ‘GPM’ is

‘Generic Package Manager’ (GPM) is my solution

- Implements ‘three different keys’ to enforce security
- Never requires a package format to change
- Uses the strategy design pattern to be ‘Generic’
- One single package-specific file on client and server
- Mirrorable using a single `rsync` command
- Already has full DEB support, Arch underway
- Investigating feasibility on Windows

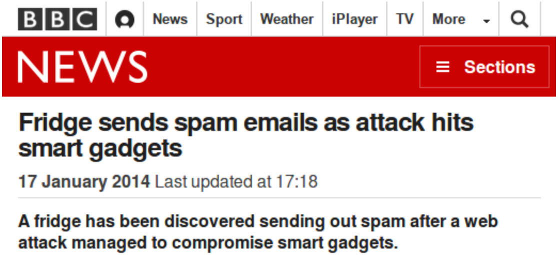
What we covered

1. Package manager 101
   - The package
   - The client
   - The repository

2. Communicating with a repository
   - Building up from first principles
   - Case study: Is that really how they do it?

3. Breaking package managers
   - Why our simple package manager is exploitable
   - How distributions keep their packages secure
   - How to bypass current package manager security

4. My Part II project
   - What ‘GPM’ is



Fridge sends spam emails as attack hits smart gadgets

17 January 2014 Last updated at 17:18

A fridge has been discovered sending out spam after a web attack managed to compromise smart gadgets.

“More than 25 percent of the [email] volume was sent by things that were not conventional laptops, desktop computers or mobile devices; instead, the emails were sent by everyday consumer gadgets such as ... connected multi-media centers, televisions and at least one refrigerator.”

Use SSL if you trust these companies

A-Trust
Actalis
AdCF
AOL
AS SK
Buypass
CA Disig
Camerfirma
CATCert
Certicámara SA
Certigna
Certinomis
certSIGN
Chunghwa Telecom
CNNIC
Comodo
ComSign
D-TRUST
DigiCert
e-Guven EBG
e-tugra
EDICOM
Entrust
GlobalSign
GoDaddy
HARICA
IdenTrust
Izenpe SA
JCSI
KEYNECTIS
Microsec
NetLock
Nets DanID
PROCERT
QuoVadis
RSA Security
S-TRUST
SECOM
StartCom
Swisscom
SwissSign
Symantec-TrustCenter
Symantec-Verisign
T-Systems
Taiwan-CA
TeliaSonera
Trend Micro
Trustis
Trustwave
TurkTrust
Unizeto Certum
Verizon Business
VISA
Web.com
Wells Fargo
WISeKey

Use SSL if you trust these companies and governments

A-Trust
Actalis
AdCF
AOL
AS SK
Buypass
CA Disig
Camerfirma
CATCert
Certicámara SA
Certigna
Certinomis
certSIGN
Chunghwa Telecom
CNNIC
Comodo
ComSign
D-TRUST
DigiCert
e-Guven EBG
e-tugra
EDICOM
Entrust
GlobalSign
GoDaddy
HARICA
IdenTrust
Izenpe SA
JCSI
KEYNECTIS
Microsec
NetLock
Nets DanID
PROCERT
QuoVadis
RSA Security
S-TRUST
SECOM
StartCom
Swisscom
SwissSign
Symantec-TrustCenter
Symantec-Verisign
T-Systems
Taiwan-CA
TeliaSonera
Trend Micro
Trustis
Trustwave
TurkTrust
Unizeto Certum
Verizon Business
VISA
Web.com
Wells Fargo
WISeKey

- Government of France
- Hong Kong Post Office, Government of Hong Kong
- Japanese Ministry of Internal Affairs and Communications
- Government of Spain, ACCV
- Government of Taiwan, Root Certification Authority
- The Netherlands' PKIoverheid7
- Government of Turkey, Kamu SM

Certificate Authorities do get hacked

Mid-2009	Many CAs caught missing crucial check Adding null byte to certificate tricked most browsers
15th Mar, 2011	Comodo partner hacked 9 certificates generated, including Google and Mozilla
10th Jul, 2011	DigiNotar hacked 531 or more certificates stolen, including '*'
Aug 2011	TURKTRUST were idiots Accidentally issued CA certs rather than SSL certs Wasn't noticed until 25th Dec, 2012
2012	Verizon-CyberTrust issued TNB a 512-bit EV cert Valid for 2 years, crackable in 73 CPU days