Welcome to

8. Secure Systems Design and Implementation

KEA Kompetence Computer Systems Security 2019

Henrik Lund Kramshøj hlk@zencurity.com @kramse

Slides are available as PDF, kramse@Github

8-secure-systems-design.tex in the repo security-courses

Plan for today

Subjects

- Principle of least privilege, fail-safe defaults, separation of privilege etc.
- Files, objects, users, groups and roles
- Naming and Certificates
- Access Control Lists
- DNSSEC

Exercises

- DNSSEC, SPF, DMARC - DNS based updates to your email domain security

Reading Summary

Bishop chapter 14: Design Principles

Bishop chapter 15: Representing Identity

Bishop chapter 16: Access Control Mechanisms

Skim, Setuid demystified

Some thoughts on security after ten years of qmail 1.0

Wedge: Splitting Applications into Reduced-Privilege Compartments

Books examples

Our book chapter 14 starts out with some examples:

Sendmail M4 macro language - compiling config files

College privacy and cheating

Prisoner communication - Attorney–client privilege

Bernstein and Woodward - 1-bit phone calls

Principle of Least Privilege

Principle of least privilege

Definition 14-1 The *principle of least privilege* states that a subject should be given only those privileges that it needs in order to complete the task.

Principle of Least Authority

Definition 14-2 The *principle of least authority* states that a subject should be given only the authority that it needs in order to complete its task.

Principle of Fail-Safe defaults

Definition 14-3 The *principle of fail-safe defaults* states that, unless a subject is given explicit access to an object, it should be denied access to that object.

Principle of Economy of Mechanism

Definition 14-4 The *principle of economy of mechanism* states that security mechanisms should be as simple as possible.

Principle of Complete Mediation

Definition 14-5 The *principle of complete mediation* requires that all accesses to objects be checked to ensure that they are allowed.

Principle of Open Design

Definition 14-6 The *principle of open design* states that the security of a mechanism should not depend on the secrecy of its design or implementation.

Mobile data encryption A5/1 key - see next page

Mobile data encryption A5/1 key

Real Time Cryptanalysis of A5/1 on a PC Alex Biryukov * Adi Shamir ** David Wagner ***

Abstract. A5/1 is the strong version of the encryption algorithm used by about 130 million GSM customers in Europe to protect the over-the-air privacy of their cellular voice and data communication. The best published attacks against it require between 240 and 245 steps. ... In this paper we describe new attacks on A5/1, which are based on subtle flaws in the tap structure of the registers, their noninvertible clocking mechanism, and their frequent resets. After a 248 parallelizable data preparation stage (which has to be carried out only once), the actual attacks can be **carried out in real time on a single PC.**

The first attack requires the output of the A5/1 algorithm during the first two minutes of the conversation, and computes the key in about one second. The second attack requires the output of the A5/1 algorithm during about two seconds of the conversation, and computes the key in several minutes. ... The approximate design of A5/1 was leaked in 1994, and the exact design of both A5/1 and A5/2 was reverse engineered by Briceno from an actual GSM telephone in 1999 (see [3]).

Source: `http://cryptome.org/a51-bsw.htm`

Principle of Separation of Privilege

Definition 14-7 The *principle of separation of privilege* states that a system should not grant permission based on a single condition.

Principle of Least Common Mechanism

Definition 14-8 The *principle of least common mechanism* states that mechanisms used to access resources should not be shared.

Principle of Least Astonishment

Definition 14-9 The *principle of least astonishment* states that security mechanisms should be designed so that users understand the reason that the mechanism works they way it does and that using the mechanism is simple.

Make documentation correct, but the program best

Files, objects, users, groups and roles

Definition 15-1 A *principal* is a unique entity. An *identity* specifies a principal.

Authentication binds a principal ro a representation of identity internal to a computer.

Example files and URLs

Users represented by user IDs

Groups and roles, sets of entities

Naming and Certificates

Naming and certificates, certificates binds cryptographic keys to identifiers

Domain Name System Security Extensions (DNSSEC)

Domain Name System Security Extensions (DNSSEC)

DNSSEC was originally specified in the following three RFCs:

RFC 4033 – DNS Security Introduction and Requirements

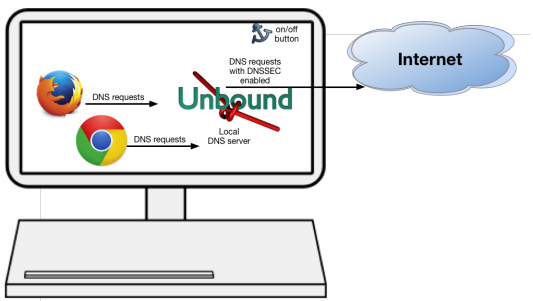
RFC 4034 – Resource Records for the DNS Security Extensions

RFC 4035 – Protocol Modifications for the DNS Security Extensions

`https://www.internetsociety.org/resources/deploy360/2011/dnssec-rfcs-3/`

Using DNSSEC we can put keys into DNS, not trusting the usual browser root CAs

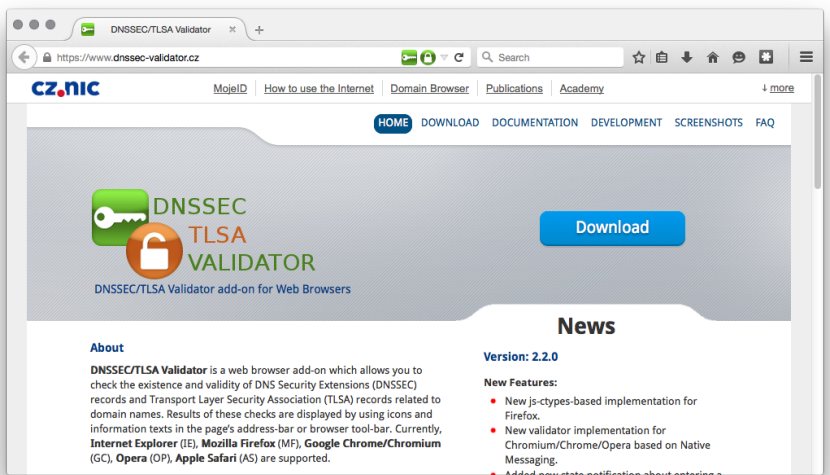
DNSSEC trigger



DNSSEC-trigger secure local DNS server for your Windows or Mac laptop.

- DNSSEC Validator for Firefox
  `https://addons.mozilla.org/en-us/firefox/addon/dnssec-validator/`
- OARC tools `https://www.dns-oarc.net/oarc/services/odvr`
- `http://www.nlnetlabs.nl/projects/dnssec-trigger/`

DNSSEC get started now



"TLSA records store hashes of remote server TLS/SSL certificates. The authenticity of a TLS/SSL certificate for a domain name is verified by DANE protocol (RFC 6698). DNSSEC and TLSA validation results are displayer by using several icons."

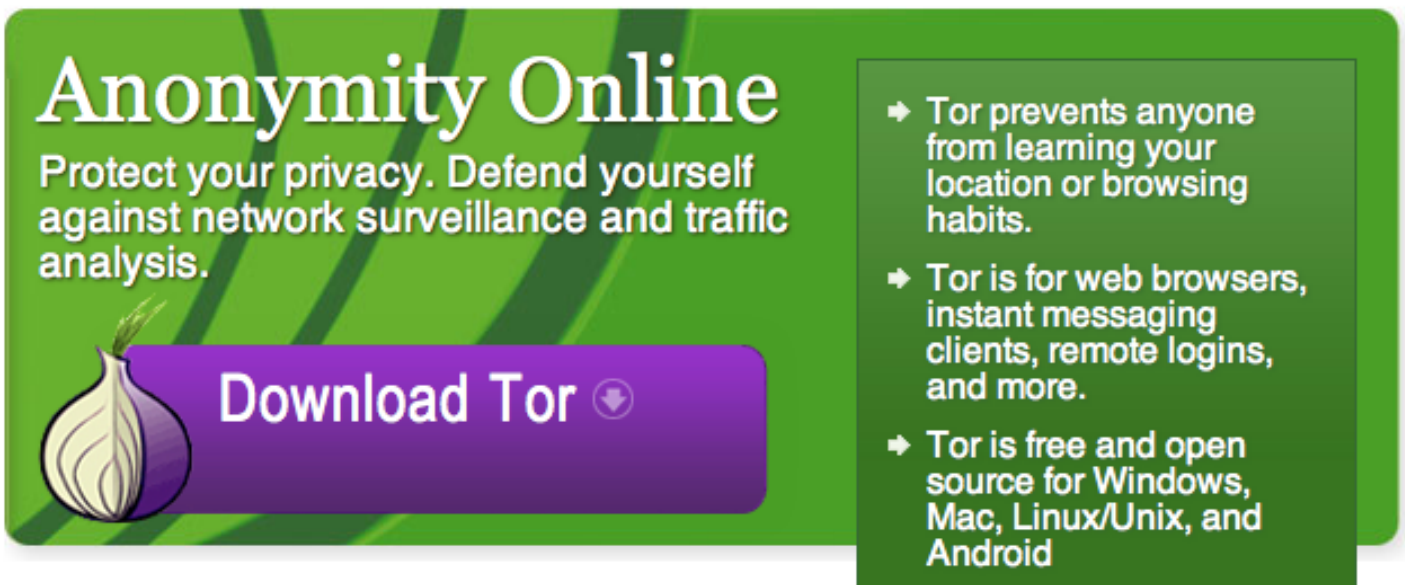
DNSSEC and DANE

"Objective:

Specify mechanisms and techniques that allow Internet applications to establish cryptographically secured communications by using information distributed through DNSSEC for discovering and authenticating public keys which are associated with a service located at a domain name."

DNS-based Authentication of Named Entities (dane)

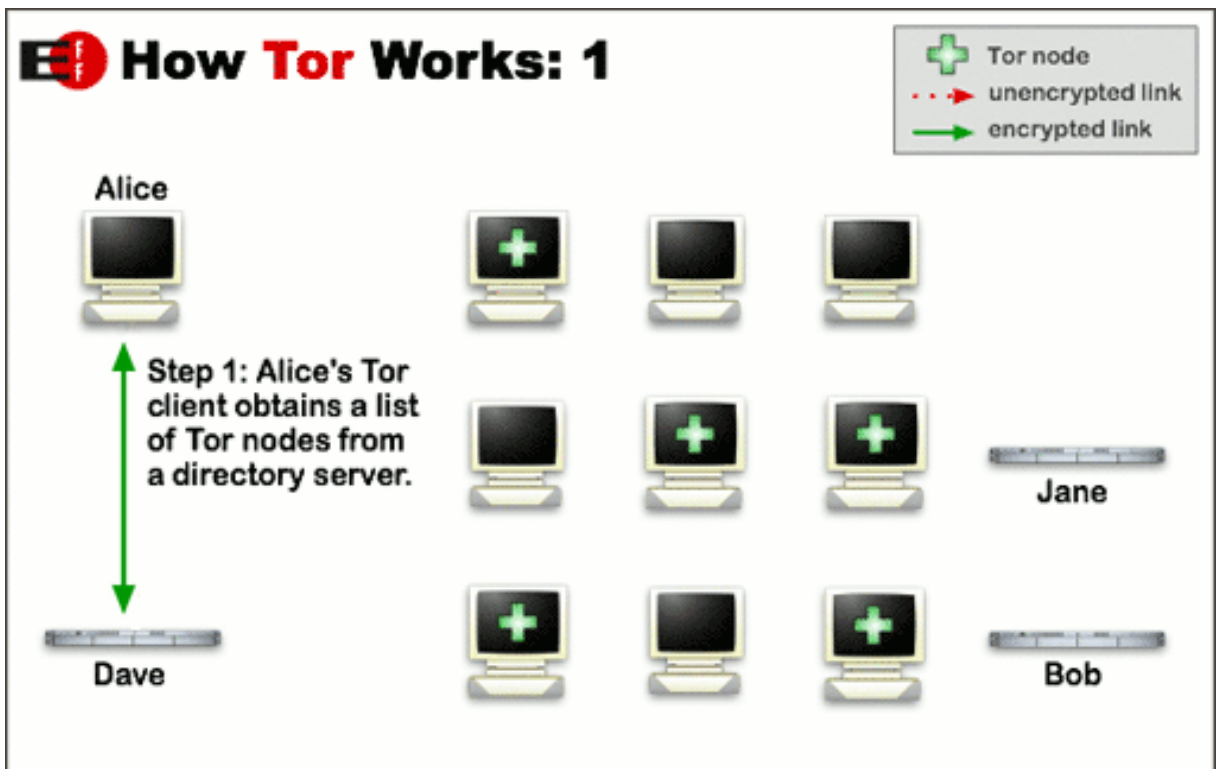
Tor project anonym webbrowsing



`https://www.torproject.org/`

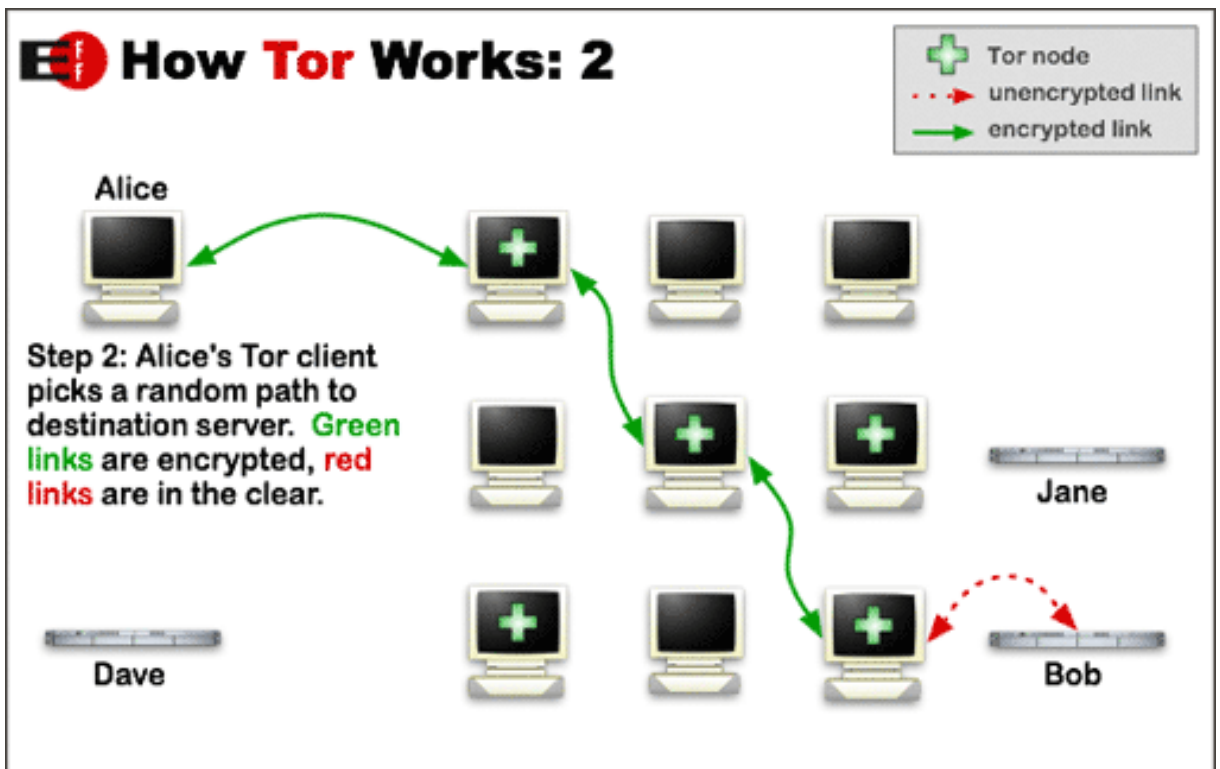
Der findes alternativer, men Tor er mest kendt

Tor project - how it works 1



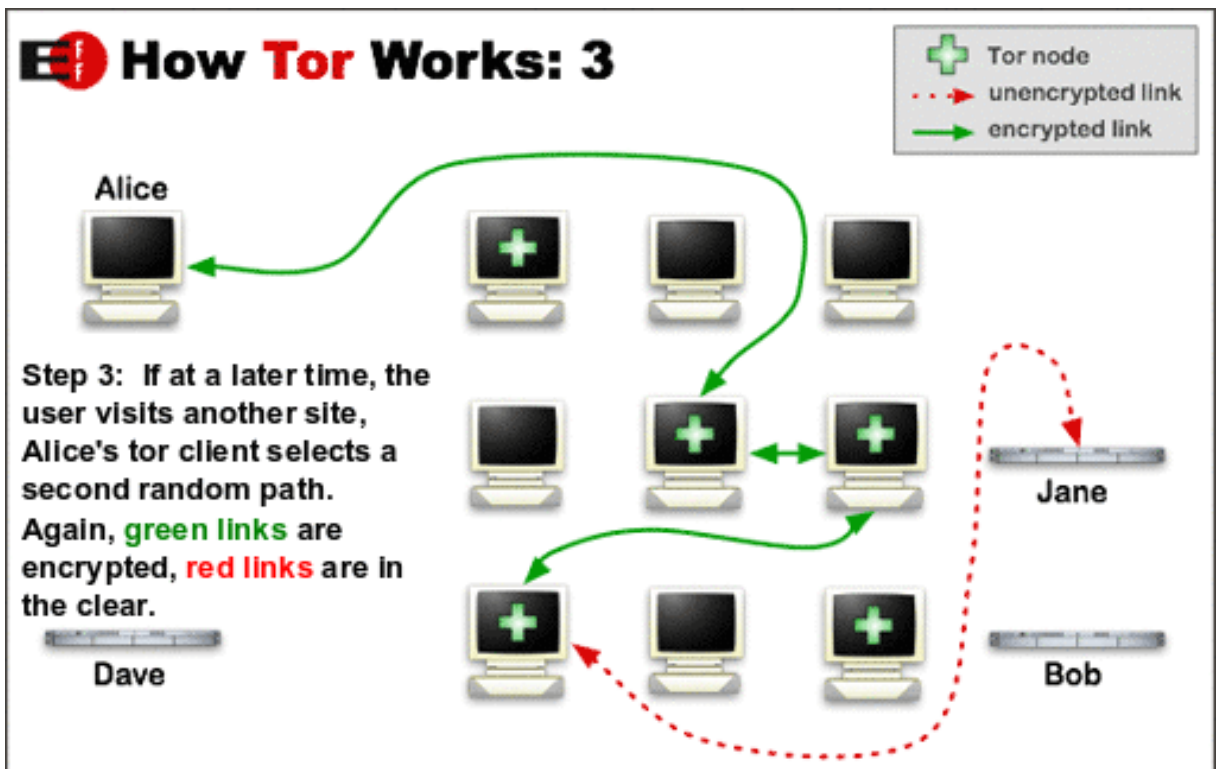
pictures from `https://www.torproject.org/about/overview.html.en`

Tor project - how it works 2



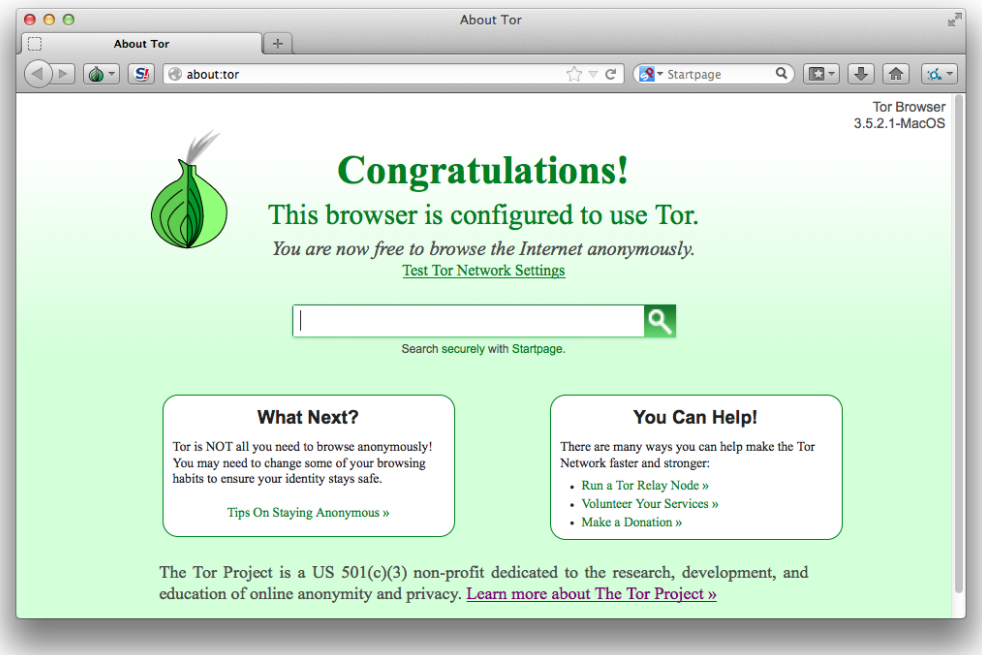
pictures from `https://www.torproject.org/about/overview.html.en`

Tor project - how it works 3



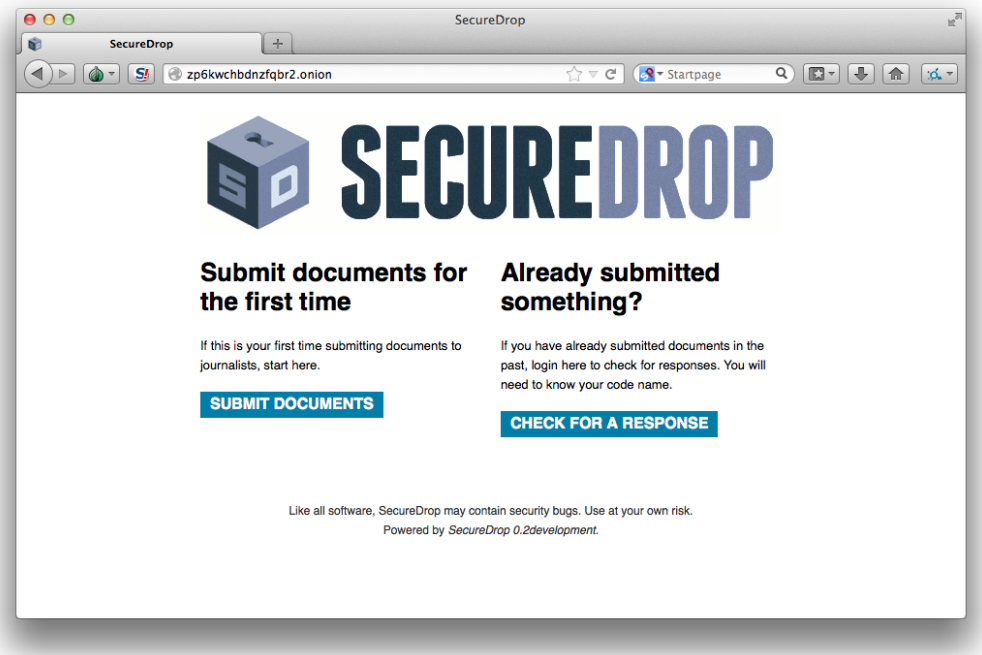
pictures from `https://www.torproject.org/about/overview.html.en`

Torbrowser - anonym browser



Mere anonym browser - Firefox i forklædning

Hidden Service web site



.onion er Tor adresser - hidden sites

`http://www.radio24syv.dk/dig-og-radio24syv/securedrop/` - now defunct

Access Control Mechanisms

Access control lists (ACL)

Used for file systems, as described in the book

Also used for firewall and router filters, network ACLs

Capabilities - used for docker in the Linux kernel

Privileges

Capsicum: practical capabilities for UNIX Robert N. M. Watson - for next time!

Ring-based access control supported by CPUs for many years
Compare kernel mode vs user mode

Setuid demystified

> Access control in Unix systems is mainly based on user IDs, yet the system calls that modify user IDs (uid-setting system calls), such as setuid, are poorly designed, insufficiently documented, and widely misunderstood and misused. This has caused many security vulnerabilities in application programs.

Setuid Demystified Hao Chen, David Wagner, and Drew Dean, Proceedings of the 11th USENIX Security Symposium, August 05 - 09, 2002

- Sometimes a user need to modify resources not owned by themselves
- Most common example is changing their password in the user database
- So while the program `passwd` runs it has the privileges of the root user, setuid-root program
- Previously Unix systems would have several 100s of setuid programs,
  today OpenBSD has less than 30 I think, and privilege seperated see OpenSSH
- Note also the many differences in Unix variants!

Setuid differences in Unix variants

setuid() Although setuid is the only uid-setting sys- tem call standardized in POSIX 1003.1-1988, it is also the most confusing one. First, the required permission differs among Unix systems. **Both Linux and Solaris** require the parameter newuid to be equal to either the real uid or saved uid if the effective uid is not zero. As a surprising result, setuid(geteuid()), which a program- mer might reasonably expect to be always permitted, can fail in some cases, e.g., when ruid=100, euid=200, and suid=100. On the other hand, setuid(geteuid()) always succeeds in FreeBSD. **Second, the action of setuid dif- fers not only among different operating systems but also between privileged and unprivileged processes.** In So- laris and Linux, if the effective uid is zero, a successful setuid(newuid) call sets all three user IDs to newuid; oth- erwise, it sets only the effective user ID to newuid. On the other hand, **in FreeBSD a successful setuid(newuid) call sets all three user IDs to newuid** regardless of the effective uid.

Setuid Demystified Hao Chen, David Wagner, and Drew Dean, Proceedings of the 11th USENIX Security Symposium, August 05 - 09, 2002

This is reality, and very confusing.

Setuid example CVE-2018-14665

The three required commands, Hickey said, are:

```
cd /etc; Xorg -fp
"Root::16431:0:99999:7:::" -logfile
shadow  :1;su
```

Source: Matthew Hickey, cofounder of security firm Hacker House

- The X11 Window System is often setuid root
- Requires access to screen memory, keyboard, mouse etc.
- Not the only problem found in X11 over the years, incomplete list at:
  `https://www.cvedetails.com/vulnerability-list/vendor_id-88/product_id-147/X.org-X11.html`

Formal verification

> Fortunately, we can note that there is a lot of symme- try present. If we have **a non-root user ID, the behav- ior of the operating system is essentially independent of the actual value of this user ID**, and depends only on the fact that it is non-zero. For example, the states (ruid, euid, suid) = (100, 100, 100) and (200, 200, 200) are isomorphic up to a substitution of the value 100 by the value 200, since the OS will behave similarly in both cases (e.g., setuid(0) will fail in both cases).

Setuid Demystified Hao Chen, David Wagner, and Drew Dean, Proceedings of the 11th USENIX Security Symposium, August 05 - 09, 2002

- The Setuid Demystified paper moves on to a formal model, but Reality bites again: `https://thehackernews.com/2018/12/linux-user-privilege-policykit.html`
- *Red Hat has recommended system administrators not to allow any negative UIDs or UIDs greater than 2147483646 in order to mitigate the issue until the patch is released.*
- `\fliptable` everything is insecure

Qmail Security

> The qmail security guarantee In March 1997, I took the unusual step of publicly offering $500 to the first person to publish a verifiable security hole in the latest version of qmail: for example, a way for a user to exploit qmail to take over another account. My offer still stands. Nobody has found any security holes in qmail. I hereby increase the offer to $1000.

Some thoughts on security after ten years of qmail 1.0, Daniel J. Bernstein

- Started out of need and security problems in existing Sendmail
- Bug bounty early on. Donald Knuth has similar for his books

Qmail Security Paper, some answers

- Answer 1: eliminating bugs $->$ Enforcing explicit data flow, Simplifying integer semantics, Avoiding parsing
- Answer 2: eliminating code $->$ Identifying common functions, Reusing network tools, Reusing access controls, Reusing the filesystem
- Answer 3: eliminating trusted code $->$ Accurately measuring the TCB, Isolating single-source transformations, Delaying multiple-source merges, Do we really need a small TCB?

Qmail vs Postfix

> I failed to place any of the qmail code into untrusted prisons. Bugs anywhere in the code could have been security holes. The way that qmail survived this failure was by having very few bugs, as discussed in Sections 3 and 4.

Some thoughts on security after ten years of qmail 1.0, Daniel J. Bernstein

- This is NOT a comlete comparison of Qmail and Postfix `http://www.postfix.org/`!
- Postfix is comprised of many processes and modules. These modules typically are also chrooted and report back status only through very restricted interfaces
- It is also possible to turn off many components, allowing the system run with less code
- No Postfix program is setuid, all things are run by a master control process. A small setgid program used for mail submission - writing into the queue directory

Source: being a Postfix user and *Secure Coding: Principles and Practices* Eftir Mark Graff, Kenneth R. Van Wyk, June 2009

Wedge Reduced-Privilege Compartments

We present Wedge, a system well suited to the splitting of complex, legacy, monolithic applications into fine-grained, least-privilege compartments. Wedge consists of two synergistic parts: OS primitives that create compartments with default-deny semantics, which force the programmer to make compartments' privileges explicit; and Crowbar, a pair of run-time analysis tools that assist the programmer in determining which code needs which privileges for which memory objects.

Wedge: Splitting Applications into Reduced-Privilege Compartments Andrea Bittau, Petr Marchenko, Mark Handley, Brad Karp NSDI'08 Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, San Francisco, California — April 16 - 18, 2008

Pledge, and Unveil, in OpenBSD

Compare to Pledge, and Unveil, in OpenBSD

- Applies to multiple different sorts of programs, privsep, privdrop unpriviledged
- Illegal operations crash the program. (SIGABRT)
- Pledge: Realistic subsets of POSIX functionality
- The pledge system call forces the current process into a restricted-service operating mode
  `https://man.openbsd.org/pledge.2`
- Ping pledges “stdio inet dns” - only need these, no read,write,create-path need to access file system!
- Unveil limit filesystem access. Many very simple: unveil(“/dev”, “rw”)
- The first call to unveil removes visibility of the entire filesystem from all other filesystem-related system calls (such as open(2), chmod(2) and rename(2)), except for the specified path and permissions.
  `https://man.openbsd.org/unveil.2`

Source: man-pages and
`https://www.openbsd.org/papers/BeckPledgeUnveilBSDCan2018.pdf`

Hardenize - web sites with testing

Multiple sites provide testing of domains and configurations

`https://www.hardenize.com/`

`https://internet.nl/`

`https://observatory.mozilla.org/` - try www.zencurity.dk which fails

`https://www.ssllabs.com/`

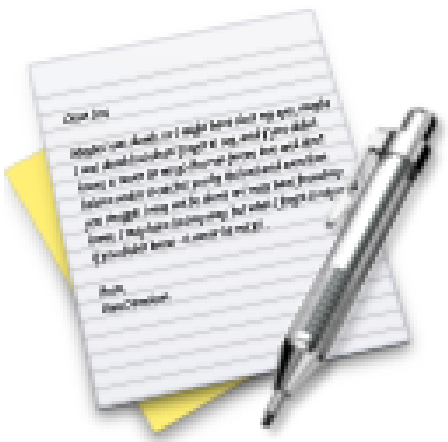
`https://securityheaders.com/`

`https://webbkoll.dataskydd.net/en`

Now we will check our own sites, and create plans.

Which parts are most interesting to you? Not having your domains abused for spamming or headers and security of your own web sites?

Exercise

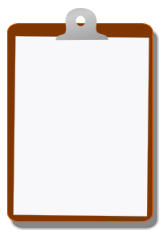


Now lets do the exercise

Email Security 2019 up to 45min

which is number **19** in the exercise PDF.

For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books
Most days have less than 100 pages, but some days may have more!

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools