Availability and Security

Choose any One

Peter Gutmann

University of Auckland

Availability vs. Security

Security

- In case of any issues, raise the alarm and shut things down

Availability

- In case of any issues, keep going at any cost

Availability is more a concern in IT services

Dependability is more a concern in SCADA/embedded

- This talk will use both interchangeably

Availability concerns dictate that in the case of a problem the system allows things to continue

- Security concerns dictate that in the case of a problem the system doesn’t allow things to continue

Case Study: System Power

Availability concerns can be a powerful motivator

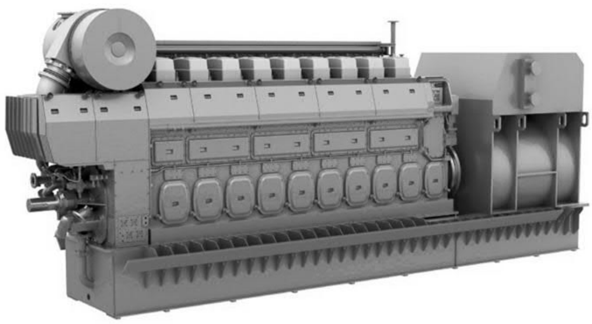
Data centre was built with marine diesel generators for backup power

- Readily available in arbitrary sizes/power levels
- Marine diesels also come with a built-in cooling system
- That would be "the ocean"

Source: Thinglink

Case Study: System Power (ctd)

Less concern about marine diesels overheating than conventional generators



Source: DirectIndustry

- Makes them more compact than standard generators
- Space was a concern for the data centre in question

Case Study: System Power (ctd)

The data centre wasn't anywhere near the ocean

Used a stand-in consisting of a large water cistern whose contents were flushed through the generators' cooling systems

Source: ClimateTech

- When the cistern had emptied, the generators' thermal cut-outs shut them down

5

Case Study: System Power (ctd)

Management's response to this was to have the safety interlocks on the generators disabled

- Might get an extra five or ten minutes out of them
- Could potentially ride out a power outage that they wouldn't otherwise have survived

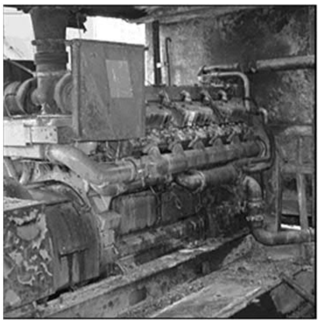
Source: Axon

6

Case Study: System Power (ctd)

Preferable to run the generators to destruction than to risk having the data centre go down

- You won’t find this in the MCSE or CCNA training material



Source: Artzat

7

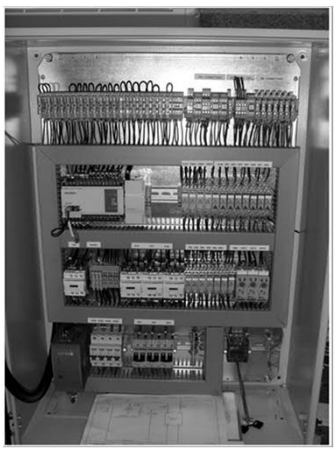
Dependable Systems

Theory of dependable systems…

Dependable systems can experience faults

- A fault doesn’t necessarily reduce the dependability of a system

Fault must result in an error in order to cause a problem



Source: Sparkfun

8

Dependable Systems (ctd)

If the error propagates beyond a system barrier so that it becomes visible to the rest of the system, it becomes a failure

> A fault can manifest itself as an error […] and the error can ultimately cause a failure
>
> — ISO 26262, “Road Vehicles — Functional Safety”

Dependable Systems (ctd)

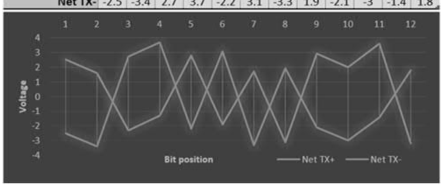
Only the full progression fault → error → failure is a visible problem

- Handled via fault detection, isolation, and recovery (FDIR)

Example from computer networking

- Fault: Electrical glitch induced onto Ethernet cable
- Error: Corrupted data packet
  - Detection: Failed CRC/FCS check
  - Isolation: Packet dropped
  - Recovery: Kicked upstairs, typically TCP-level
- Failure: None, fault mitigated

Induced Noise:	0	-0.9	0.2	1.2	0.3	0.6	-0.8	-0.6	0.4	-0.5	1.1	-0.7
Bits:	1	1	0	0	1	0	1	0	1	1	1	0
Net TX+	2.5	1.6	-2.3	-1.3	2.8	-1.9	1.7	-3.1	2.9	2	3.6	-3.2
Net TX-	-2.5	-3.4	2.7	3.7	-2.2	3.1	-3.3	1.9	-2.1	-3	-1.4	1.8



Source: Practical Networking

Dependability

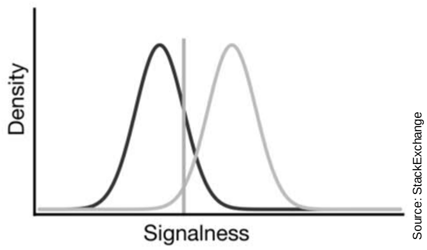
Fault mitigation strategies

- Is the value within a range of plausible values?
  - Vehicle engine temperature, speed, etc
  - Unless the vehicle is powered by a Mr.Fusion, an engine temperature of 3000°C is suspect
- Is the combination of values within a range of plausible values?
  - Engine speed / vehicle speed / gear ratio
- Do multiple redundant sources agree?
  - Angle-of-attack sensors on aircraft
- Exotic rigorous solutions
  - Predictor/corrector models like Kalman filters

Dependability (ctd)

Signal metrics

- Signal quality
- Timestamps
- Sequence numbers
- Signal-changed status



Timing protection

- Protecting from activities that take too long to complete
- Excessive runtime upsets response-time guarantees for other components

Mitigations

Substitute values

- If a value is implausible, substitute an approximation for use in subsequent calculations
- Malfunctioning sensor, use last known good value

Voting / redundancy

- 2oo3 or similar mechanisms

Liveness monitoring of subsystems

- Watchdogs, heartbeats

Diverse monitoring

- External monitor ensures the system remains within safety bounds

Mitigations (ctd)

Execution sequence monitoring

- Check control flow graph (CFG)
- Monitor control flow through basic blocks
  - As a convenient side-effect, severely hampers ROP
  - A rare case of dependability *and* security

Reliability trumps everything

- “Limp home” mode as a design safe state
- Disable some subsystems, e.g. keep ABS (anti-lock braking) but no ESC (electronic stability control)
- c.f. MEL in aircraft
  - Minimum (functioning) equipment list for an aircraft to be considered airworthy

Fault-Tolerance

Not just a fancy name, the system is literally tolerant of faults

- A great deal of engineering effort goes into providing this capability

Overreacting to faults can actually be harmful

> In some situations taking recovery actions due to errors […] may cause more damage than it does good. Reacting to such errors may cause an over-reaction where the recovery actions may put the system in a state where it is less safe than previously
>
> — "Explanation of Error Handling on Application Level", AUTOSAR

Fault-tolerance is the diametric opposite of what crypto/security does

Fault-Intolerance

In crypto/security, the goal is to find the single bit that's out of place

- One single bit out of place → fail

```
a. If the length of L is greater than the input
   limitation for the hash function (2^61 - 1 octets
   for SHA-1), output "decryption error" and stop.

b. If the length of the ciphertext C is not k octets,
   output "decryption error" and stop.

c. If k < 2hLen + 2, output "decryption error" and
   stop.
```

— PKCS #1 v2.1

- "… and stop" means "fault and error and failure" all in one

Fault-Intolerance (ctd)

Once you’ve found the discrepancy, you’ve won

Source: LA Times

No known standard covers how to continue after this point

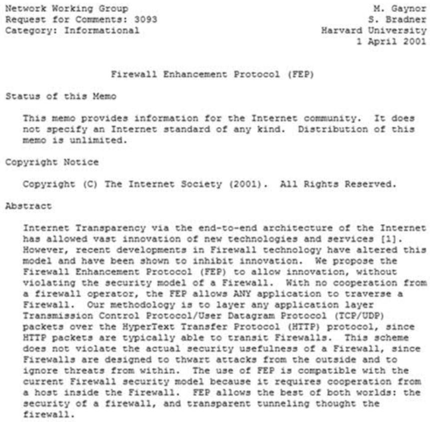
- c.f. vast literature on fault tolerance and error recovery

Security vs. Availability at the Design Level

Any RFC ever

… the server MUST NOT … the client MUST NOT …

More common is
to leave it unspecified



Network Working Group — M. Gaynor
Request for Comments: 3093 — S. Bradner
Category: Informational — Harvard University
1 April 2001

Firewall Enhancement Protocol (FEP)

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

Abstract

Internet Transparency via the end-to-end architecture of the Internet has allowed vast innovation of new technologies and services [1]. However, recent developments in Firewall technology have altered this model and have been shown to inhibit innovation. We propose the Firewall Enhancement Protocol (FEP) to allow innovation, without violating the security model of a Firewall. With no cooperation from a firewall operator, the FEP allows ANY application to traverse a Firewall. Our methodology is to layer any application layer Transmission Control Protocol/User Datagram Protocol (TCP/UDP) packets over the HyperText Transfer Protocol (HTTP) protocol, since HTTP packets are typically able to transit Firewalls. This scheme does not violate the actual security usefulness of a Firewall, since Firewalls are designed to thwart attacks from the outside and to ignore threats from within. The use of FEP is compatible with the current Firewall security model because it requires cooperation from a host inside the Firewall. FEP allows the best of both worlds: the security of a firewall, and transparent tunneling thought the firewall.

Source: IETF

Security vs. Availability at Design Level (ctd)

Try finding a statement in a standard for a protocol (TLS, S/MIME, PGP, SSH, OCSP, SCEP, CMP, …) that tells you what to do if a crypto validation fails

- Not even a “if XYZ validation fails the client MUST terminate the connection”

There’s just… nothing

- OK, one or two small notes specifically pointing out particular special-case oddball conditions

If a client receives an extension type in ServerHello that it did not request in the associated ClientHello, it MUST abort the handshake with an unsupported_extension fatal alert
— RFC 5246 / TLS

Security vs. Availability at Design Level (ctd)

You can write an implementation that ignores MAC failures, decryption errors, and invalid signatures, and be fully standards compliant

- As long as you deal with a small number of obscure corner cases specifically called out as MUST NOTs

Look at the spec for your favourite protocol after the talk

- Someone else’s problem?
- It was never even considered?
- The experienced driver will usually know what’s wrong?

Fault Mitigation vs. Security

Plausibility checks: Binary yes/no

Execution sequence monitoring: No

Substitute values: No

Voting/redundancy: No (except in Type 1 crypto hardware)

Liveness checks/signal metrics: N/A

Timing protection: Public-key crypto operations are variable-time

- Some operations like keygen only terminate probabilistically

Fault Mitigation vs. Security (ctd)

Continuing with degraded functionality

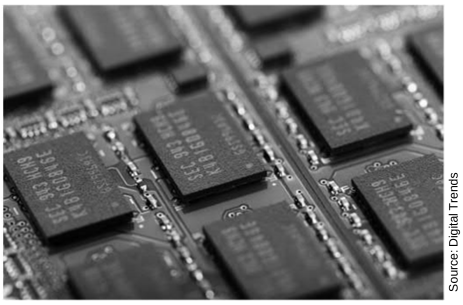
Case Study: Memory Leaks

RTOS has a memory leak problem

- Chief software engineer: “Of course it leaks!”

Solution

- Calculate the worst-case memory usage (including leaks)
- Provision the hardware with double that amount of memory



Arrghhh!!!!!!!!!!!!

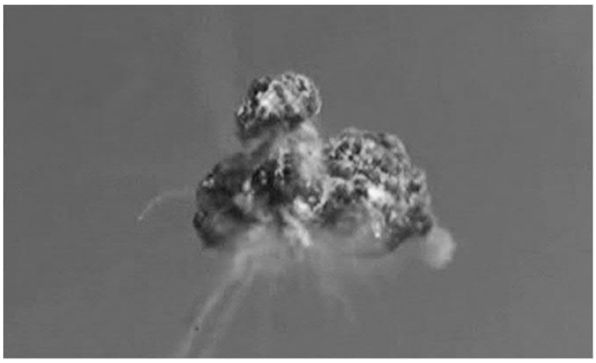
Case Study: Memory Leaks (ctd)

This memory leak was in a critical real-time system

- The OS for a missile

Case Study: Memory Leaks (ctd)

Only needed to run once, after which it was garbage-collected



Source: NationStates

This was a perfectly sensible way of dealing with the leaks

Case Study: Memory Leaks (ctd)

Alternative method for dealing with leaks and similar slow degradation issues: Rejuvenation

"We have a memory leak/loss of sensor sensitivity/ degradation of measurement accuracy, …"

- Measure/calculate performance degradation as 5% per day
- Calculate worst-case timing for sufficient loss of functionality that the system is affected, i.e. it becomes a failure
- Schedule the system to restart before then

Rejuvenation is a standard fault-mitigation technique in critical control systems

- You don't need to fix the fault, just mitigate it

Availability vs. Security

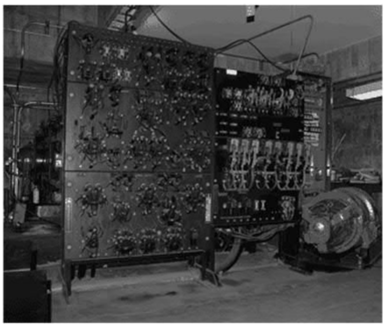
High-availability systems (e.g. SCADA) cannot go down

- Ever

MTBF requirements of ten years are not uncommon

- May be run for decades

Often can't be patched or updated



Source: Eletec Global

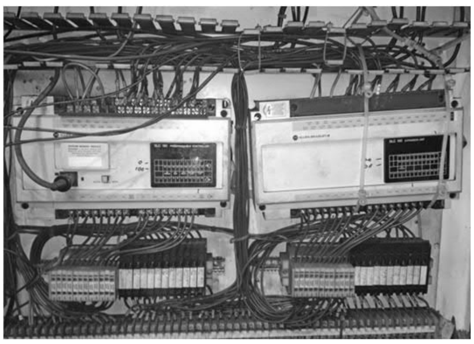
Availability vs. Security (ctd)

These systems are incredibly long-lived

- Trying to apply current security technologies to them is difficult

There are PLCs currently in active use that might be expected to run TLS, but that predate

- TLS
- SSL
- Web browsers using SSL
- The companies that make the web browsers
- The web itself



Source: Eletec Global

Availability vs. Security (ctd)

Problem: CA-issued certificates are valid for one year

- MTBF 12 months ≪ MTBF 10 years or more
- Ignore certificate expiry
  - In any case it's just a CA billing mechanism
  - Certificate that's perfectly fine on day n doesn't become completely insecure on day $n + 1$
- Issue your own certificates with infinite lifetimes

Problem: Certificates may suddenly stop working due to revocation

- Ignore CRLs and OCSP

Availability vs. Security (ctd)

Problem: Devices don't have DNS names, or even fixed IP addresses

- Identity = address often doesn't hold in any case
- Device might be identified by an IEEE EUI64 ID (OUI + unique ID)
- ID the device by EUI64 or similar after you connect

The more checking you do, the greater the chance of something breaking

- Disable the ~~safety interlocks~~ nuisance checks to make sure things keep running

Availability vs. Security (ctd)

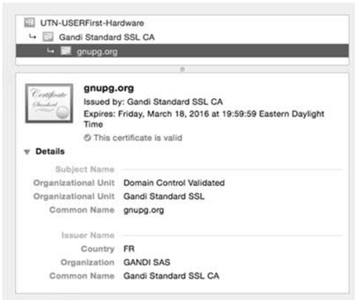
Having a reactor control system shut down because its certificate has expired is seen as something of a liability

When PLCs' certificates expire, they just disappear off the network. Plus, 99 percent of the industrial world has no idea what a certificate is, so how do they troubleshoot the problem at 2am?

— "Control Systems Security from the Front Lines"

To ensure it works, ignore ID info, expiry dates, and revocations

- That's the entire certificate except for the key



Availability vs. Security (ctd)

Standard practice for the military is to turn off crypto when things go hot

- Availability is more important than security
- By the time the enemy has intercepted, processed, and acted on cleartext tactical comms, it's too late to do much with the information

Source: Eletec Global

Availability vs. Security (ctd)

Air Force has a similar issue

- (US) Air Force study for the European tactical air environment indicated that their vulnerabilities from losing comms (due to jamming) were greater than those from unencrypted comms

Source: Wikipedia

- Senior Air Force officer said that he needed an anti-jam capability (i.e. availability) so badly "he would trade aircraft for it"

Availability vs. Security (ctd)

This issue was explicitly recognised in the Ware Report, 1970

> In a military command and control system where delay can mean disaster, operational urgency may dictate that a calculated risk of unauthorized divulgence be assumed in order to maintain continued service to users
>
> — "Security Controls for Computer Systems", Willis Ware

Case Study: Hardcoded Passwords

Hardcoded passwords are bad

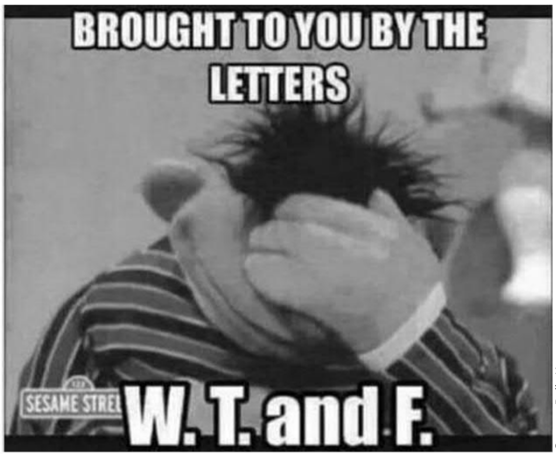
Endless security vulnerabilities due to hardcoded credentials

Case Study: Hardcoded Passwords (ctd)

Vendor advertises that the hardcoded passwords in their devices are more secure than the hardcoded passwords used by their competitors

Source: Twitter

Case Study: Hardcoded Passwords (ctd)



Source: Daily Kos

Think about that one for a minute…

37

Case Study: Hardcoded Passwords (ctd)

This makes sense in critical-infrastructure applications like power grid control

- Repair crew locked out of a system for lack of a password is a far bigger problem than cyberbogeymen
  - Too little access to a system is vastly worse than too much
- If the default password is hard to guess then a random scanning attack won't get in
- It would require a specific, targeted attack, not an automated scan

Using a complex hardcoded password is a genuine feature since it's more secure than using a simple one

38

Case Study: Hardcoded Passwords (ctd)

[Pause to allow collection of scattered brain fragments]

Source: Twitter

It’s Security Jim, but Not as we Know It

The term “security” as used with control systems is very different from the way that security people use it

- Often refers to reliable communications rather than what cryptographers would think of as security

Security measures include

- CRC checks
- Message sequence numbering
- Requirement to receive multiple trigger messages in order to initiate a high-consequence event
- Redundant encoding of control messages to make sure that a particular command really is what was intended
- …

It’s Security Jim, but Not as we Know It (ctd)

Security means not flipping a 240MW generator in and out of circuit due to improper signalling, not something involving digital signatures or firewalls

Source: Moscow Times

It’s Security Jim, but Not as we Know It (ctd)

Ultimate case of this is the weak link/stronglink design used for managing nuclear weapons triggering

Weak link prevents initiating the detonation

> Shown to predictably fail prior to the barrier or strong links losing their integrity in the event of catastrophically severe environments which would eventually breach the exclusion region barrier to the warhead or the strong links
>
> — “Designing and Building Nuclear Weapons to Meet High Safety Standards”

It’s Security Jim, but Not as we Know It (ctd)

Stronglink controls initiation of the detonation

The two strong links are in series and are of different designs, in order to minimize the risk of common-mode failures. One strong link is operated by human intent […] The second strong link must receive unique preprogrammed features of the designated trajectory of the weapon system during delivery to the target
— “Designing and Building Nuclear Weapons to Meet
High Safety Standards”

This technology is called ENDS (enhanced nuclear detonation safety)

It’s Security Jim, but Not as we Know It (ctd)

Designed to minimise the chances of accidental triggering

- Conceptually, requires navigating through a multi-level tree making exactly the right choice at each fork to initiate a detonation

If it receives a wrong signal in the event of an accident, or due to hostile action, it will lock up the system and no arming signal can be transmitted
— “Designing and Building Nuclear Weapons to Meet …”

Strong links require an enabling input different from any electrical, mechanical, or environmental stimuli produced by exposure to an abnormal environment or accident
— “Designing and Building Nuclear Weapons to Meet …”

It’s Security Jim, but Not as we Know It (ctd)

Created in response to B-47/B-52 accidents that came close to triggering detonations

- Most notorious case was the Goldsboro accident in 1961
- One bomb went through almost all the steps of the arming sequence, with only one or two minor measures preventing detonation (reports vary)

Source: Twitter

Diggers found the ARM/SAFE switch. It was in the ARM position
— “Orange resident recalls holding future in his hands”

It’s Security Jim, but Not as we Know It (ctd)

How do we design a trigger mechanism that’s as immune to false triggering as possible

- Even remote possibilities have to be accounted for due to the high stakes

Solution: Unique signal generator

The purpose of a unique signal (UQS) in a nuclear weapon system is to provide an unambiguous communication of intent to detonate […] in a manner that is highly unlikely to be duplicated or simulated in normal environments and in a broad range of ill-defined abnormal environments. Thus, the UQS serves both a reliability function and a safety function
— “The Unique Signal Concept for Detonation Safety in Nuclear Weapons”

It’s Security Jim, but Not as we Know It (ctd)

Initial designs looked at how electrical signalling could be done in a manner that wasn’t subject to false triggering

- What if a nuclear-armed bomber crashed and wiring was exposed in the wreckage?
- Power cable swinging back and forth across it creates a pulse train

Source: Twitter

Calculate a statistically unlikely signal and require that as part of the stronglink process

It’s Security Jim, but Not as we Know It (ctd)

OK, if you really need to know…

- A sufficient number of events: 24 for a single-try (not remotely resettable) device, and many more for a multiple-try (remotely resettable) device (47 were used in the MC2969, but this number is device-dependent and application-dependent).
- Event-wise balanced (as nearly as possible equal numbers of 'A' type events and 'B' type events).
- Pair-wise balanced (as nearly as possible equal numbers of 'A A', 'A B', 'B A', and 'B B' pairs).
- No more than four 'A's or 'B's together.
- Different numbers of groupings (singles, pairs, etc.) of 'A's and 'B's.
- Minimal length of repeated strings, including complements and reverse order.
- Non-periodic.
- Non-symmetrical.
- Minimum length of strings repeated in all other patterns used for UQSs, with particular attention to strings aligned in the same position.

It’s Security Jim, but Not as we Know It (ctd)

Control systems security is implemented through a more limited version of this style of design

Differential signalling on all circuits

- True = 0 + 1, False = 1 + 0
- Both lines powered (short circuit) or both lines unpowered → fault
- Signal is present statically (circuit latch-up) rather than dynamically (driven by a clock pulse) → fault

It’s Security Jim, but Not as we Know It (ctd)

Clock pulse isn’t necessarily a standard CPU clock

- Capacitively-coupled AC signal that acts as an additional gating signal
- Applying DC power to the circuit won’t produce an output

Passive (through-hole) components may be mounted by having the circuit traces they’re soldered to run on opposite sides of a double-sided circuit board

- Mitigates short circuits due to something falling across them

It’s Security Jim, but Not as we Know It (ctd)

Logic is implemented with

- Redundant buses
- Self-checking logic designs
  - AND gate can be implemented as $a \wedge b$ and $\neg a \vee \neg b$ and cross-checked
- Majority-logic decoding
- Whole books full of exotic design techniques

It’s Security Jim, but Not as we Know It (ctd)

Programming is also pretty alien

- PLCs don’t run a program in any conventional sense
- Run scan cycles and are programmed in ladder logic or more modern forms like instruction list (IL), most of which boil down to fancy ladder logic
- Scan cycle takes input from various sensors, processes it, and sends out signals to control devices like plant machinery
- PLC operates in a tightly-coupled loop with the physical systems that it monitors and controls, responding to external stimuli and using internal components like counters and timers to control external physical devices

Applying conventional notions of security is… interesting

It’s Security Jim, but Not as we Know It (ctd)

Leads to odd definitions of “security”

- German term “Informationssicherheit” = Information + Sicherheit = Information Security
- “Sicherheitsgerichtete Echtzeitsysteme” = Security-oriented Real-time systems or Security for Real-time systems

Source: Springer

It’s “security” in the other sense, i.e. how to make real-time systems highly reliable

- No mention of cryptography or anything similar

Case Study: It is Good that No-one Knows

In medicine, saving the patient’s life overrides all other issues

- Messing around with computer authentication gets in the way of this

Typical example: Surgeon’s interns and residents do all of the surgeon’s computer work

- Surgeon’s time is far too valuable to spend messing around with data entry
- (Patient records are now accessible to anyone who works for or with the surgeon)

Case Study: It is Good that No-one Knows (ct

More general solution: The first person who comes in in the morning logs on

- Everyone else uses this person's logon to do their work
- Most successful deployment of single sign-on ever

Source: Wisegeek

Case Study: It is Good that No-one Knows (ct

Variant: The person with the greatest level of access does the logging on

Variant: The lowest-status person gets tasked with going around everyone's computers hitting the space bar to prevent auto-logout

System designers and managers may not even know their assumptions do not hold — and may make policy and other decisions based on fantasy. These rule-bending scenarios may, in fact, be closer to general operating procedures than are often recognised. The unreal views of the system designers remain untouched by feedback

— "Healthcare Information Technology's Relativity Problems"

Case Study: It is Good that No-one Knows (ct

When hospitals strictly enforce login policies, the impact is catastrophic

- Clinician at a UK hospital estimated he spent 1½ hours of his 14-hour day just logging on
  - (To get a hundred-hour week, you work 14-hour days)
- UK allocated £40M just to try and reduce logon times to medical IT systems

It is frankly ridiculous how much time our doctors and nurses waste logging on to multiple systems
— UK Health Secretary Matt Hancock

Case Study: It is Good that No-one Knows (ct

Variant of this single sign-on method is used with banks that require that traders log out at the end of the day

- Traders hired interns with clipboards containing everyone's passwords to go around before trading opened and log in all the traders
- Like surgeons, traders' time is valuable
- More single sign-on in action

Source: Trading Room Lisboa

Case Study: It is Good that No-one Knows (ct

Another variant of this appears in grid computing

- One project member obtains a magic certificate to make things work and everyone else shares it
- Site security manager had to complain to users that although they didn’t mind everyone using the same cert, its original owner had been dead for some time and could they please have someone generate a new one to share around

Grid computing security is often handled through short-lived certificates

- Do the same thing as Kerberos tickets, but badly
- Rope in grad students to continuously renew certs for staff members who have long-running grid jobs

Case Study: It is Good that No-one Knows (ct

A rare few institutions have actually addressed the problem

Air traffic control uses shared (group) passwords for systems that control radar, navigation and communications gear, the instrument landing system (ILS), and similar equipment

Source: Aerotime Hub

- This is by design, not as a workaround
- Everyone in the group knows the password
- Something won’t be inaccessible for lack of a password
- 41% of FAA facilities in the US use group passwords

Case Study: It is Good that No-one Knows (ct

Another institutionalised practice is the 24-hour logon

- No-one ever logs out
- Used in high-availability systems that can't risk inaccessibility due to lack of a password
- Speeds up shift changes in continuously-manned systems

A variant is the immortal certificate (see earlier slides)

- Certificate has an infinite lifetime, or
- Certificate details are never checked

Case Study: It is Good that No-one Knows (ct

Problem comes about from the 1960s mainframe model for security

- Walk into the (singular) physically secured terminal room
- Sit down at one of the terminals
- Access the (singular) mainframe

Source: Wikipedia

Case Study: It is Good that No-one Knows (ct

Has never been updated for modern use

- Nomadic system/device use
- Work is collaborative/shared rather than single-user
- Conventional access-control models don't work

This is yet another of the many reasons for the enduring ubiquity of passwords

- None of this critical-to-availability functionality would work with other authentication mechanisms

Case Study: Random Numbers

Random number generation

- `/dev/random` blocks until enough entropy is available
- `/dev/urandom` doesn't

Tinfoil-hat response

- Only ever use `/dev/random`

Linux kernel provides a system call `getrandom()`

- In the kernel for years but wasn't supported in glibc
- Google "ulrich drepper"
- Some support finally added in late 2016

Case Study: Random Numbers (ctd)

Blocks, but also uses `/dev/urandom`

- Worst of both worlds

`getrandom()` → `make_application_hang_at_random()`

- Quite literally so

Two years after this issue was first pointed out…

> Openssh taking minutes to become available, booting takes half an hour... because your server waits for a few bytes of randomness
>
> ```
> [    4.428797] EXT4-fs (vda1): mounted filesystem with
> ordered data mode. Opts: data=ordered
> [ 130.970863] random: crng init done
> ```
>
> Systemd makes this behaviour worse

- Just wanted to point that out…

Case Study: Random Numbers (ctd)

Nearly a year after *that*…

> It began to be common for Embedded Linux systems to "get stuck at boot“ […] Over time, the issue began to even creep into consumer-level x86 laptops […] It can therefore be argued that there is no way to use getrandom() on Linux correctly

Patch submitted to make it nonblocking as a user-settable option

- Followed by an interminable thread stretching to hundreds of messages arguing about it

Case Study: Random Numbers (ctd)

If you disabled the blocking, what would happen?

- (Your application wouldn't appear to hang/crash at random any more)
- "Somewhere on the Internet there may be a system that may be running with reduced entropy"
- How is that exploitable by an attacker?

67

Wicked Problems

So why is this so hard?

Source: Wkikpedia

This (and many other issues) are examples of wicked problems

- Concept from the field of social planning
- Proposed in the 1970s as a means of modelling the process for dealing with social, environmental, and political issues

68

Wicked Problems (ctd)

Amongst a wicked problem's weaponry are such diverse elements as…

Lack of any definitive formulation of the problem

Lack of a stopping rule

- One of the core requirements for dealing with a wicked problem becomes not deciding too early which solution you're going to apply

Solutions that are rateable only as "better" or "worse" and not true or false

- Particularly bad for security geeks
- There are only two options, absolutely secure or absolutely insecure

Wicked Problems (ctd)

No clear idea of a which steps or operations are necessary to get to the desired goal

A variety of ideological and political differences among stakeholders

> The difference between them is simple: [algorithm design] is 'hard science'. [Security] is 'people wanking around with their opinions'
> — Linus Torvalds, 2007

Wicked Problems (ctd)

A wicked problem presents...

- No clear idea of what the problem is
- No clear idea of how to get to a solution
- No easy way to tell whether you've reached your goal or not
- All of the participants are pulling in different directions

71

Case Study: High-performance Sports Cars

Fit a more powerful engine

- Adds extra weight
  - Slows it down again
- Adds size
  - If taken to extremes leaves little room for anything else, including a driver

72

Case Study: High-performance Sports Cars (c

Reduce weight by fitting a lighter engine

- Have to make the car lighter to compensate for the less powerful engine

If taken to extremes leads to a car that's little more than an exoskeleton with a motorcycle engine

- Has limited appeal to the general market

Case Study: High-performance Sports Cars (c

Use exotic materials like carbon fibre to decrease weight

- Raises the price and again discourages buyers

Case Study: High-performance Sports Cars (c

Strip out as many weight-adding features as possible

- Trade-off between performance and comfort
- Some jurisdictions have safety regulations that affect what you can and can't do
- Tradeoff between being able to sell the car in a particular market and making performance-reducing changes

Case Study: Audio Woo-Woo

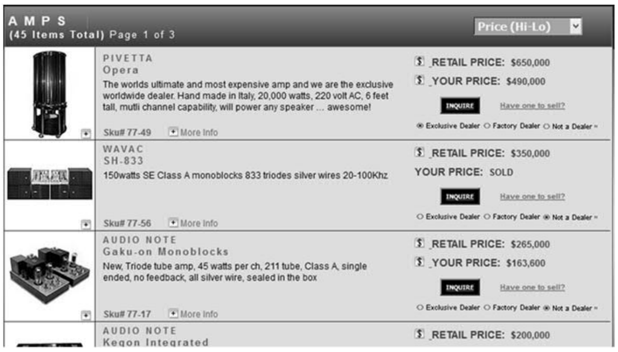
High-end audio is like high-performance sports car design, only much sillier

Case Study: Audio Woo-Woo (ctd)

OK, that's not really true…

Only stopping rule is "how much money does ~~the sucker~~ the customer have?"

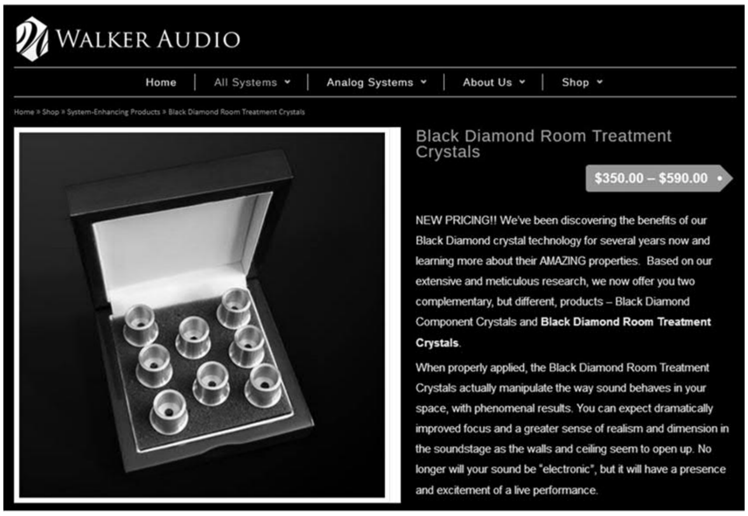
- Any solution *you* sell is better than what everyone else has (by definition)



Source: Higher-Fi

Limits are defined by how much woo-woo you can come up with

Case Study: Audio Woo-Woo (ctd)



Anything goes…

Case Study: Audio Woo-Woo (ctd)

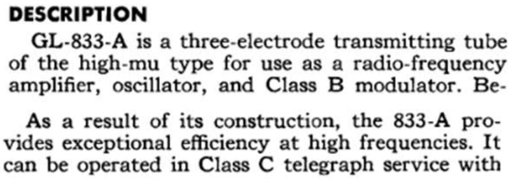
Of course we'd never go for this in the security field…

Case Study: Audio Woo-Woo (ctd)

- $30,000 iPod dock demo'd at CES 2012
- Behringer iNuke Boom car-sized dock

Case Study: Audio Woo-Woo (ctd)

Example: Wavac SH-833 ($350,000 amp from earlier slide) using 833 tubes



DESCRIPTION

GL-833-A is a three-electrode transmitting tube of the high-mu type for use as a radio-frequency amplifier, oscillator, and Class B modulator. Be-

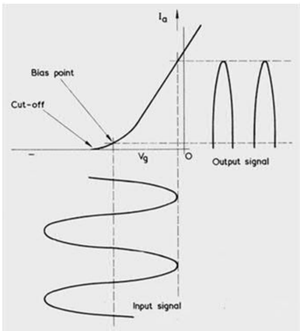
As a result of its construction, the 833-A provides exceptional efficiency at high frequencies. It can be operated in Class C telegraph service with

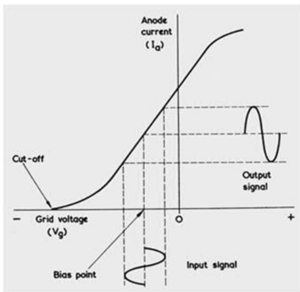
Source: eBay

- 1938 vintage 300W RCA radio transmitting tube
- Popular in many types of tube amplifiers, not just the Wavac
- Sells for ~$100-200 per tube

Case Study: Audio Woo-Woo (ctd)

Design use: Class B or C RF modulator/power amp





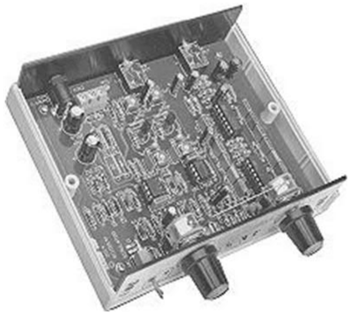
Source: National Valve Museum

Wavac use: Class A audio amp

Why pound a screw with a wrench when you can use a spanner?

Case Study: Audio Woo-Woo (ctd)

If you really want to get the genuine valve sound, you buy or build something to add all the distortion back in on a standard amp



Source: National Valve Museum

Case Study: Audio Woo-Woo (ctd)

Pair that with a generic 300W amp for $40-100…

Source: eBay

Case Study: Crypto Woo-Woo

There are equivalents to this in crypto…

Wireless USB (WUSB)

- Short-range, low-power communications

Source: FTDI

In 2004:

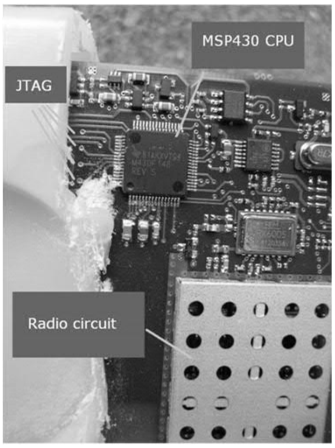
- 4096-bit DH!
- 3072-bit RSA!
- SHA-256!
- AES-CCM!
- Did we miss out anything else we could throw in?

Implementing it on $0.15 chip is Someone Else’s Problem

Case Study: Crypto Woo-Woo (ctd)

“Smart” meters

- Digital signatures!
- X.509 certificates!
- CRLs!
- The whole PKI shebang!



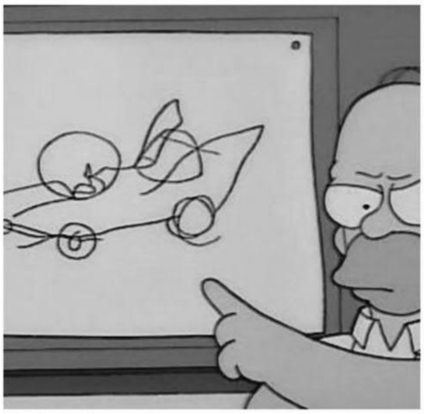
Source: rdist

MSP430F148 CPU

- 8 MHz 16-bit CPU
- 16-bit multiplier as external functional unit (no divide)
- 2kB RAM, 48kB flash
- Additional analog/digital circuitry for a power meter

You can guess how much PKI this actually implements…

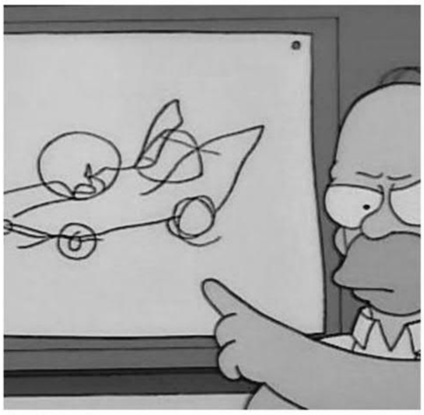
Getting Back to Sports Cars



Source: StarsAndCars

87

Getting Back to Sports Cars (alt)



88

Wicked Problems

This perfectly illustrates the characteristics of a wicked problem…

No definitive formulation of what's required for a sports car

No stopping rule to tell you that you've definitely reached your goal

- Running out of money is one oft-encountered stopping rule

The various options can only be rated in terms of tradeoffs against each other

continues…

89

Wicked Problems (ctd)

…continued

It's not obvious which steps are the best ones to take in getting to your goal

All manner of externalities

- Participants' opinions of which option is best
- Bikeshedding comes as an automatic built-in

Source: Top Gear

- Externally-applied materials and regulatory constraints on what you can and can't do

90

Conclusion