Everything is (still) broken.

Looking back at 20 years of hacking

Threat 2023

Dr. Fabian Yamaguchi - CTO Whirly Labs (Pty) Ltd

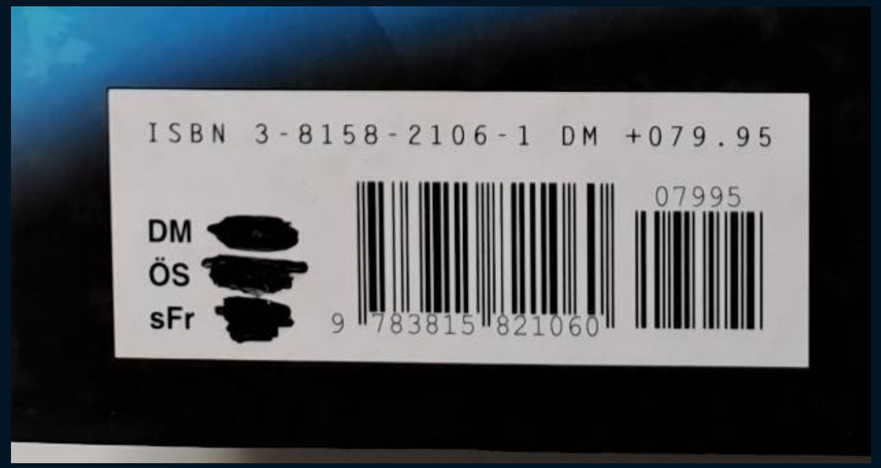
whirlylabs.com

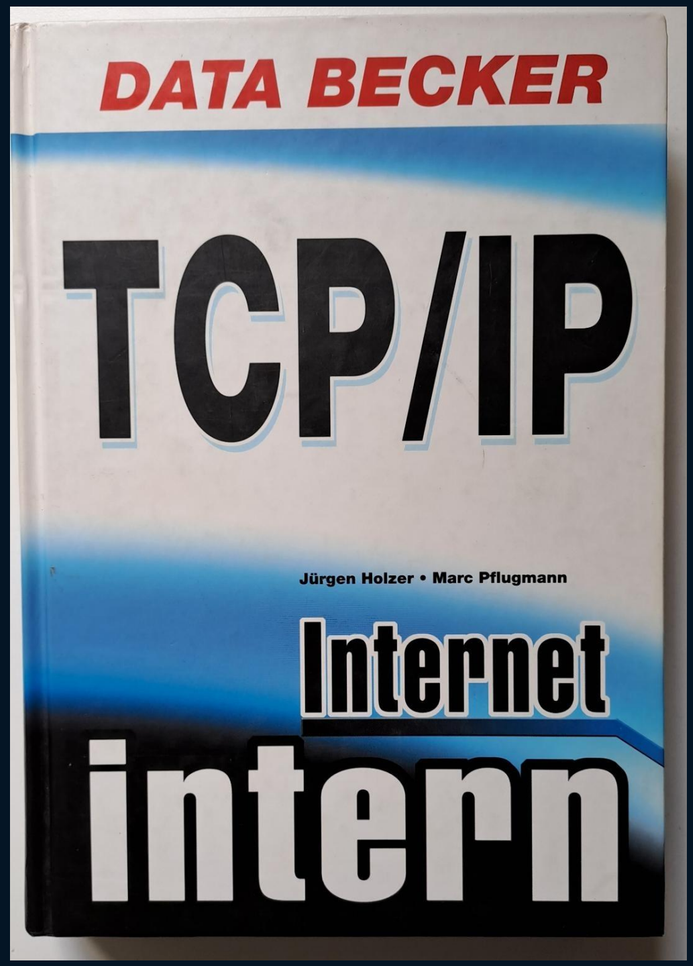
I love Hacking

- Got into hacking at the innocent age of 12, pushing 40 these days.
- First “white hat hacker” job at age 19
- Did my master thesis on hacking
- Did my PhD on hacking
- Spent a few years building automated code analysis to help in my hacking
- Today:
  - I run an IT security consultancy to show our customers how they, too, can be hacked :)
  - I teach hacking at the university to pass on the joy to tomorrow’s hackers

Late 1990's book on the internet

- Got this book from a friend on my 12th birthday
- The company that published this book no longer exists
- The currency it was bought in no longer exists
- **All of the content is still entirely relevant to how the internet functions today**

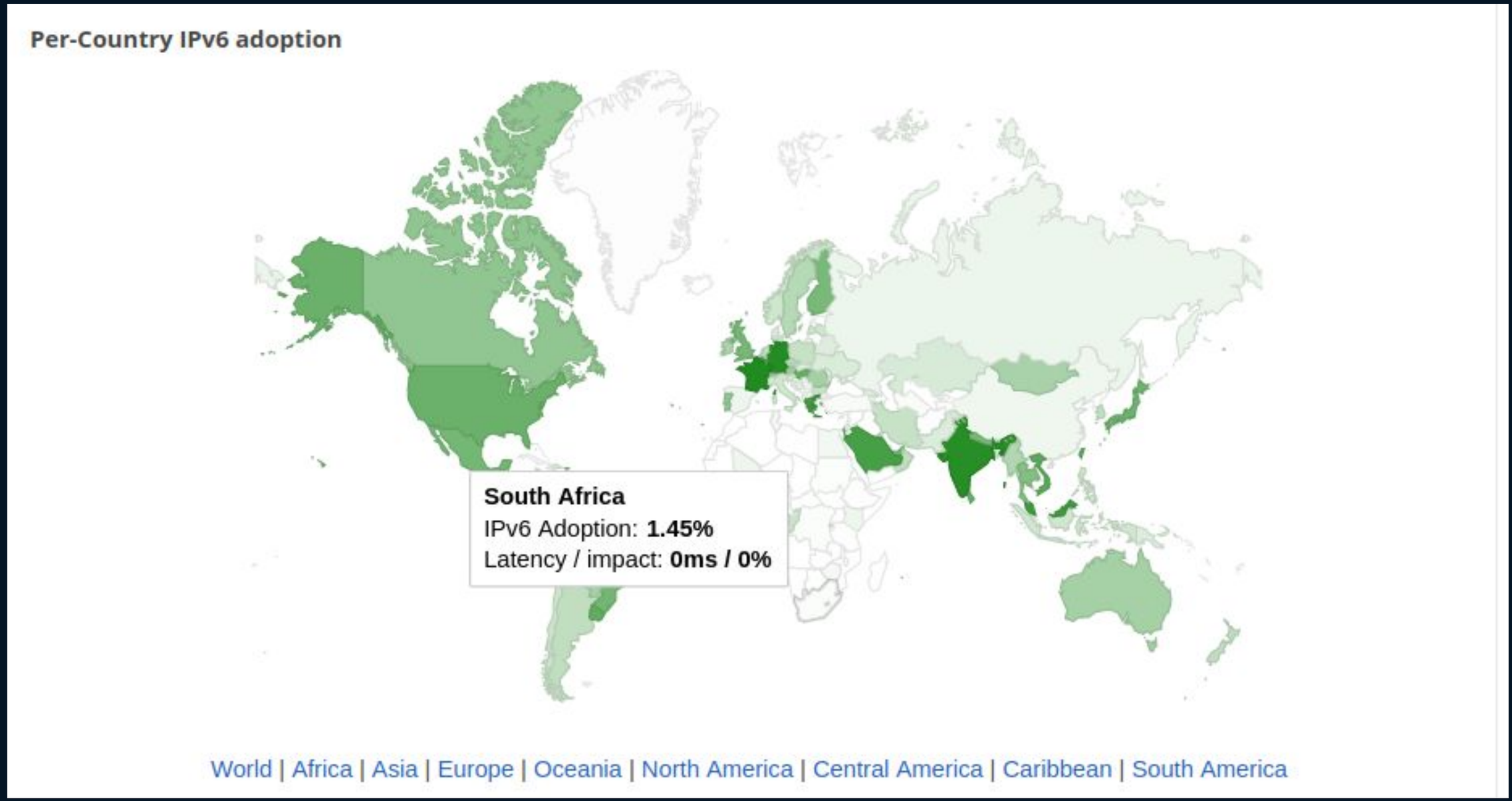




The TCP/IP(v4) stack still forms the basis of the internet today

- The protocols were devised in the 1970s
- It was believed at the time that an improved version of IPv4 named IPv6 would soon replace IPv4
- Many people back then were saying that gaining a too deep knowledge of any particular technology does not make sense because it would be out the door tomorrow.

IPv6 adoption today



What does that mean for internet security?

- The internet is based on protocols designed in the 70s without security in mind, and that is hard to change now
- Even the effort to replace IPv4 with IPv6 initiated over 20 years ago has not yet succeeded
- So, what happens when design flaws are found in internet protocols?

The “Kaminsky Bug” (2008)

- A story that illustrates how hard it is to fix anything in TCP/IP
- Dan Kaminsky presented a realistic and practical attack on the internet’s “domain name system” (part of the TCP/IP stack)
- DNS is the protocol that tells your browser where to find “fnb.co.za”
- The attack allowed attackers on the public internet to redirect users to their own fake versions of “fnb.co.za” (or any other domain!), **without seeing the communication between DNS servers**

Ingredients of the vulnerability

- The domain name system uses raw **“User Datagram Protocol (UDP)” Datagrams** (part of the TCP/IP stack)
- **Senders are not verified: you can pretend that the packet is coming from someone else (“spoofing”)**
- **The DNS protocol itself also does not provide any authentication:**
  - Your answer will be considered valid if you send it to the right place and with the right sequence number
  - The sequence number is not meant to be a secret and was often just obtained by increasing a counter
  - The sequence number was only 16 bit wide (65536 possible values), so even guessing it was easy
- Dan Kaminsky put these ingredients together and showed how they allowed to easily guess the sequence number before a valid answer from the actual server was returned =>
  this allowed “poisoning” DNS to point to point “fnb.co.za” to an attacker-controlled IP

The sad part: solution had been readily available since 1999

- An extension to DNS called **“DNSSEC”** was completed as early as 1999
- The extension allows for digital signing of DNS records to allow verifying authenticity
- To this day, DNSSEC is only supported by a small number of DNS servers on the internet
- Even if a user decides to use DNSSEC for “fnb.co.za”, if FNB does not support it, they can’t.

DNSSEC deployment has not succeeded to this day

The fix that was deployed

- To this day, DNS does not offer authentication
- Instead, the port number is now randomized so it is harder to know where to send the packet
- In 2020, a team of researchers showed that guessing the port is also quite feasible by exploiting a side channel
- The fix for that was to make the side channel less reliable
- Again, no authentication was added to DNS, and that’s understandable, because it is a huge undertaking
- It is “fixed” now, phew.

These are the crumbling foundations of the internet. Now, let’s build more on top.

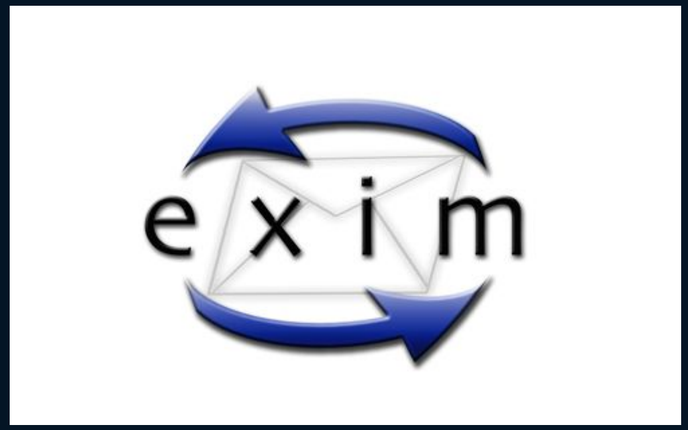
“Oh wow! Were there more devastating bugs like this?”

Yes, everything is constantly and remakably broken

- Design bugs in core protocols that are hard to fix are interesting, but the reality is that attackers care primarily about the criticality of bugs
- Dan Kaminsky's bug won a pwnie award in the category **"Most Overhyped Bug"** that year
- Reality: lots and lots of critical vulnerabilities every year, and lots of them are simply not publicly known
- In that same year (from the pwnie award winners):
  - Crafted IGMP packets allowed taking over of Windows servers
  - Debian had accidentally turned off random numbers for SSH servers
  - A large number of critical vulnerabilities were found in Wordpress
- **And today?**

CVE-2023-42115: Exim4 Pre-Auth Vulnerability

- Exim4 is the most popular e-mail server with an estimated market share of 56.78% (2018)
- Due to the vulnerability, attackers can take over exim servers without requiring any sort of authentication
- This vulnerability was announced on **September 27th, 2023**
- **By exploiting this vulnerability at scale, one can read the majority of the planet's e-mail messages**
- And e-mail to this day is still clear text!



Endless stream of vulnerabilities

- Beyond “internet killing bugs”, there is an endless stream of other vulnerabilities
- Common Vulnerability Enumeration (CVE) identifiers are assigned for publicly known vulnerabilities
- 25,029 CVE identifiers were assigned in 2022 alone
- Exponential growth of assigned CVE identifiers in the last years
- Simply managing vulnerability information has become a task in its own right

Year	2023	2022	2021	2020	2019	2018	2017	2016
Qtr4	TBA	6,231	5,200	4,387	4,822	3,614	3,570	1,590
Qtr3	6,936	6,448	5,541	4,170	5,150	4,350	4,037	1,713
Qtr2	7,134	6,365	5,005	5,011	4,091	4,613	3,612	1,775
Qtr1	7,015	6,015	4,415	4,807	3,245	3,935	3,426	1,379
TOTAL	21,085	25,059	20,161	18,375	17,308	16,512	14,645	6,457

But fear not, we have the means to enumerate them

Changes to syntax [edit]

In order to support CVE ID's beyond CVE-YEAR-9999 (aka the CVE10k problem) a change was made to the CVE syntax in 2014 and took effect on Jan 13, 2015.[11]

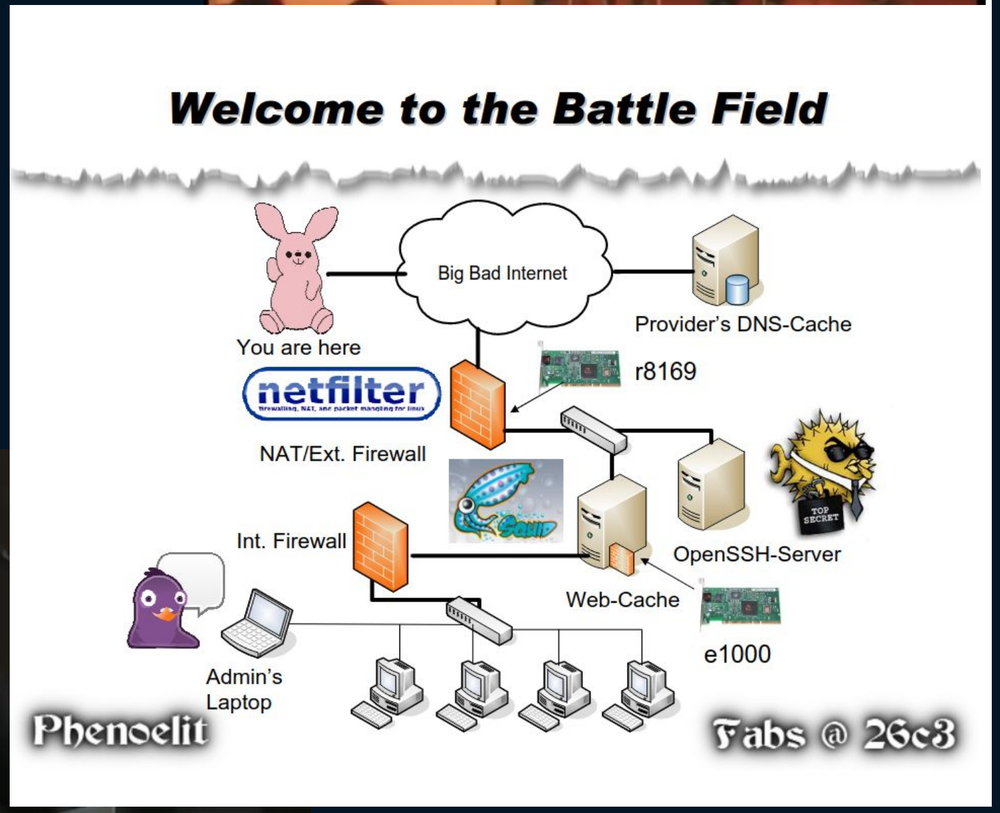
The new CVE-ID syntax is variable length and includes:

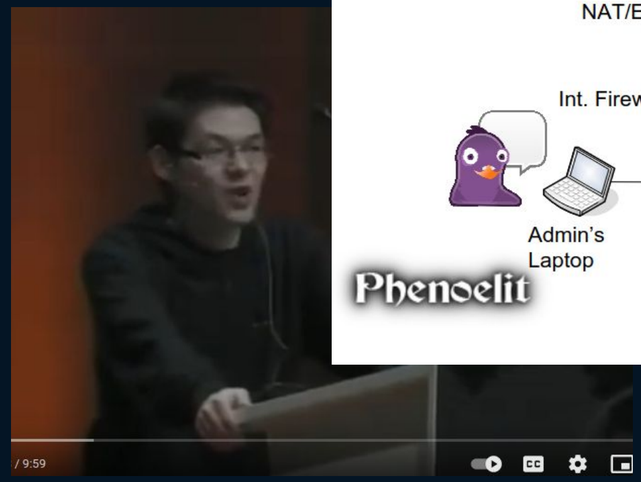
CVE prefix + Year + Arbitrary Digits

The variable-length arbitrary digits will begin at four fixed digits and expand with arbitrary digits only when needed in a calendar year; for example, CVE-YYYY-NNNN and if needed CVE-YYYY-NNNNN, CVE-YYYY-NNNNNN, and so on. This also means no changes will be needed to previously assigned CVE-IDs, which all include a minimum of four digits.

2009: cat /proc/sys/net/ipv4/fuckups

- **Key message: bugs are plentiful and you can combine them**
- Idea: drop a whole lot of vulnerabilities in a single talk at Europe’s largest hacker conference - at Christmas time
- Provide an anti thesis to the idea of the “grand bug” in favour of the many little bugs that one can combine as an attacker
- A chaotic talk on zero-day vulnerabilities in …
  - Ethernet drivers (the layer beneath TCP/IP)
  - TCP/IP and firewall implementations
  - Poisoining of proxy server caches (server side)
  - Instant messaging applications (client side)





At the edge of the network

- Off-path DNS attack is relevant when the attacker cannot monitor/modify communication between DNS servers
- If the attacker can monitor/modify it, then randomization does not matter because all the information is available
- **Immediate effect: if the attacker has control over your router/modem, then they can poison your DNS resolver with ease.**
- We did a bit of research on home plastic routers in 2021, finding them easy to fully take over
- **How did we do that?**

Hacking home routers

- With the shift to the home office, hacking home routers has been increasingly interesting
- Code quality is low, monitoring does not exist in these networks

- Asked guy from computer repair shop to get the serial port on the router working
- Promised him that he'd be on the slides
- Dumped code off of the device
- Decompiled the code
- Queried code for common vulnerabilities
- Read code for one day
- Wrote exploit for three days

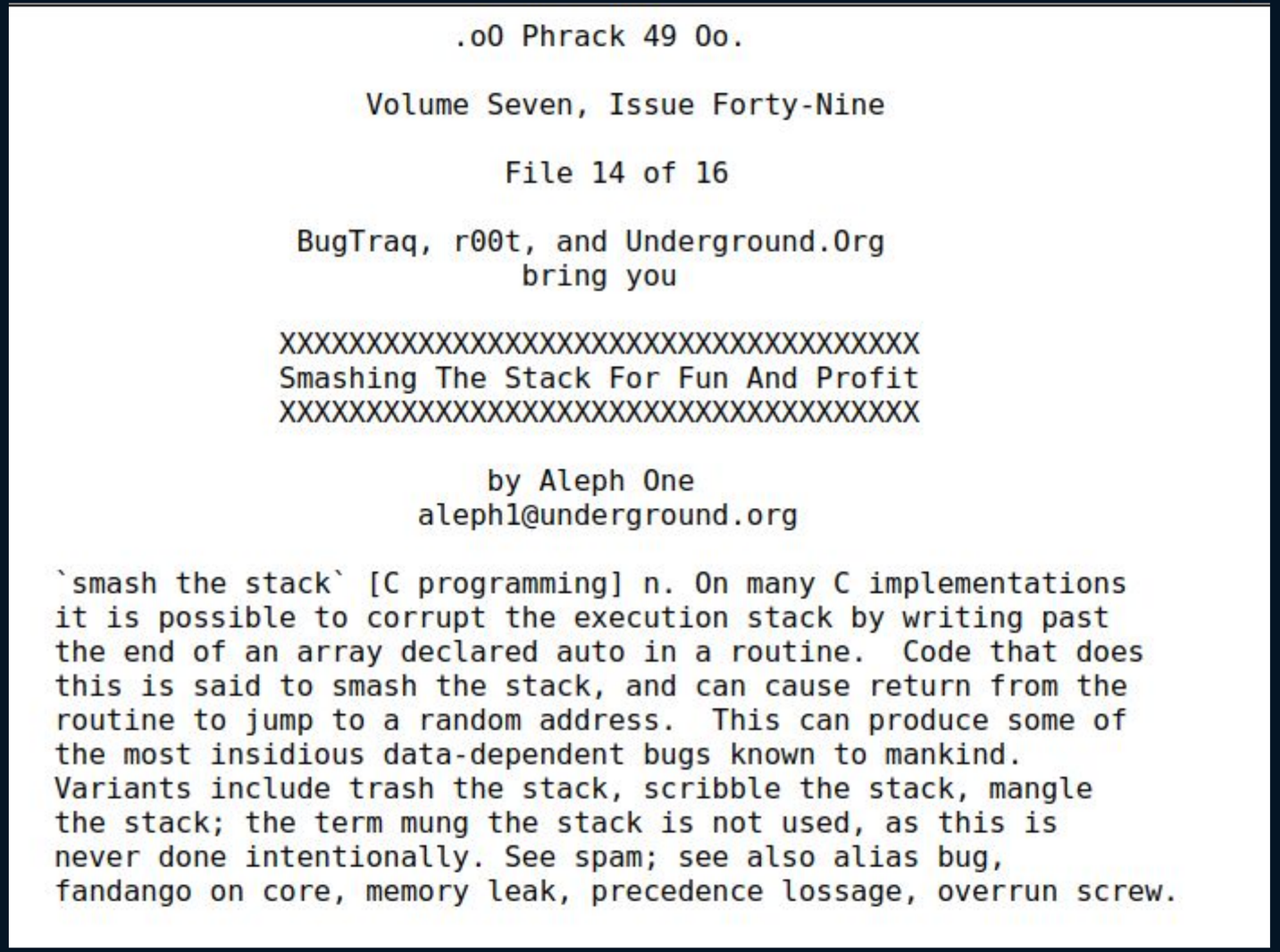
Stack-based buffer overflow

```
iVar1 = memcmp(param_3,"init ",5);
if (iVar1 == 0) {
  __nptr = (char *)memcpy(auStack168,(void *)((int)param_3 + 5),param_4 - 5);
  __nptr[param_4 - 5] = '\0';
  uVar2 = strtoul(__nptr,local_28,10);
  piVar4[1] = uVar2;
  if (((uVar2 != 0) && (local_28[0] != (char *)0x0)) && (*local_28[0] == '\0')) {
    *piVar4 = 2;
  }
  pthread_mutex_unlock((pthread_mutex_t *)&DAT_00435648);
  return 1;
}
```

- The flaw is as vanilla as it gets
- Copying of data received over the network into a buffer of static size that is too small to hold it
- **Reported to vendor: no response.**

The depressing part: smashing the stack… (1996)

- Famous paper in phrack about exploiting stack-based buffer overflow vulnerabilities is from the mid 90s!
- Memory on the program stack is corrupted, ultimately leading to execution of attacker-provided code
- Various defense mechanisms have since been deployed to make exploitation harder, and in some cases, impossible
- Yet…



.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

File 14 of 16

BugTraq, r00t, and Underground.Org
bring you

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Smashing The Stack For Fun And Profit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

by Aleph One
aleph1@underground.org

`smash the stack` [C programming] n. On many C implementations it is possible to corrupt the execution stack by writing past the end of an array declared auto in a routine. Code that does this is said to smash the stack, and can cause return from the routine to jump to a random address. This can produce some of the most insidious data-dependent bugs known to mankind. Variants include trash the stack, scribble the stack, mangle the stack; the term mung the stack is not used, as this is never done intentionally. See spam; see also alias bug, fandango on core, memory leak, precedence lossage, overrun screw.

2023: Still corrupting memory… in disbelief

- On the right: conference T-shirt (2019) for “OffensiveCon”, and entire conference centered around memory corruption
- How is this still an issue?
  - There is still a lot of C/C++ code out there in security critical components, including network equipment, browsers, operating systems
  - Again: fix exists in theory (memory safe languages for low-level programming, e.g., Rust) but adoption takes time
  - People seem to like managing their own memory (like driving without a seat belt)

What has improved: usage of SSL/TLS

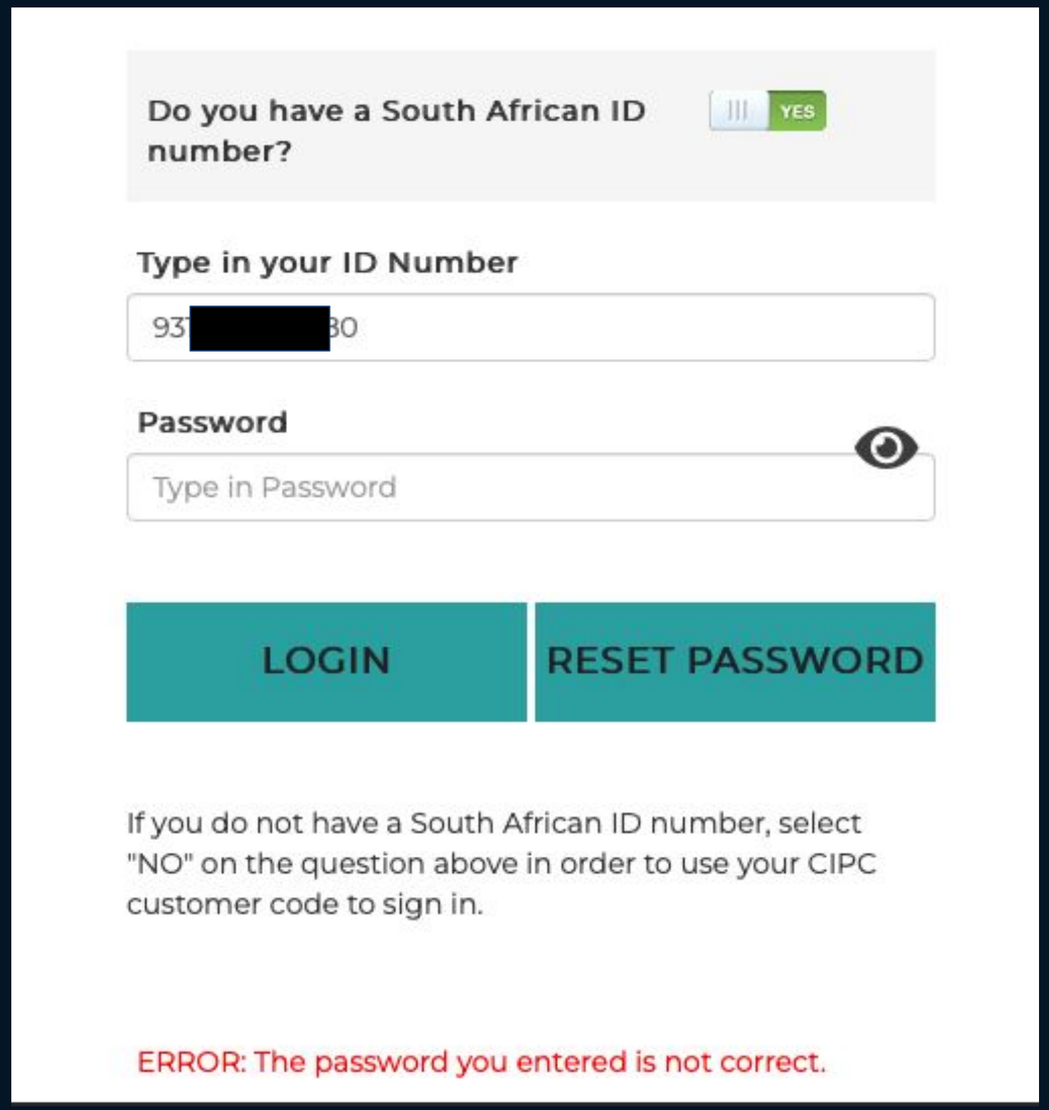
- Example: impact of DNS cache poisoning vulnerabilities are limited if **SSL/TLS** is widely deployed and certificates are validated correctly => if you do end up speaking to the wrong host, a warning is presented

- “Let’s Encrypt”-Project has significantly increased the use of SSL/TLS on the internet
- In effect, a user may still be redirected to a malicious website but that website usually does not have the valid SSL/TLS certificate

“Widely adopted” doesn’t mean “adopted where it would matter”

- Example: the government website “bizportal.gov.za” still allows HTTP (without SSL/TLS)
- In effect, it is not possible to know if you are speaking to the right website as you enter your data
- It also means that anyone on the network path can see what you enter

Side note: that particular website is a security catastrophe



Do you have a South African ID number? YES

Type in your ID Number

93[redacted]

Password

Type in Password

LOGIN RESET PASSWORD

If you do not have a South African ID number, select "NO" on the question above in order to use your CIPC customer code to sign in.

If you are a registered CIPC customer, you can proceed to login using your 13-digit South African ID number and your CIPC password.
If you are new to CIPC services, please register using your ID number.

* Please note that you might be asked verification questions related to your ID or that of your marital partner. Marital partner is the legally married partner, as per the Marriage Act.

Customers who would like to login but do not have South African ID numbers can now login using their CIPC customer code. Select "NO" on the question below the login form to use a customer code, instead of an ID number. If you don't have a CIPC customer code, then you can register for one on https://eservices.cipc.co.za/

ERROR: System.ArgumentException: Column 'agent_id_no' does not belong to table DataSet. at System.Data.DataRow.GetDataColumn(String columnName) at System.Data.DataRow.get_Item(String columnName) at BizPortal.login.btnLogin_Click(Object sender, EventArgs e) in C:\Users\admlesetja\Documents\Bizportal\login.aspx.cs:line 58

In summary

- Design vulnerabilities in core internet protocols are hard to address and are often insufficiently addressed
- Security can be built “on top” while replacing components is hard => increased complexity
- As the tech stack becomes increasingly complex, the number of vulnerabilities increases
- The potential this creates to creatively combine multiple issues to craft powerful attacks is mind blowing

- Memory corruption flaws are still a major concern, although they are harder to come by in certain well audited components, e.g., browsers, web servers.
- Exploits for these vulnerabilities have become more time consuming to produce due to mitigations
- Meanwhile, the security product industry continues to sell shiny boxes, snake oil, and compliance certificates, but that is a topic for another talk.

Enjoy the session!

Dr. Fabian Yamaguchi
CTO - Whirly Labs (Pty) Ltd
https://whirlylabs.com
fabs@whirlylabs.com
Twitter: fabsx00