black hat®
ASIA 2021
May 6-7, 2021
BRIEFINGS

THE COST OF COMPLEXITY: Different Vulnerabilities While Implementing the Same RFC

Daniel dos Santos
FORESCOUT
Active Defense for the Enterprise of Things™

Shlomi Oberman
JSOF

Who we are

FORESCOUT

Active Defense for the Enterprise of Things™

JSOF

Daniel dos Santos - RESEARCH MANAGER

Stanislav Dashevskyi - SECURITY RESEARCHER

Jos Wetzels - SECURITY RESEARCHER

Amine Amri - SECURITY RESEARCHER

Shlomi Oberman - CEO

Moshe Kol - SECURITY RESEARCHER

Outline

- **Introduction**
- **NAME:WRECK** – Breaking DNS implementations
- **Impact**
- **Mitigation** – Fixing DNS implementations
- **Conclusion**

Introduction

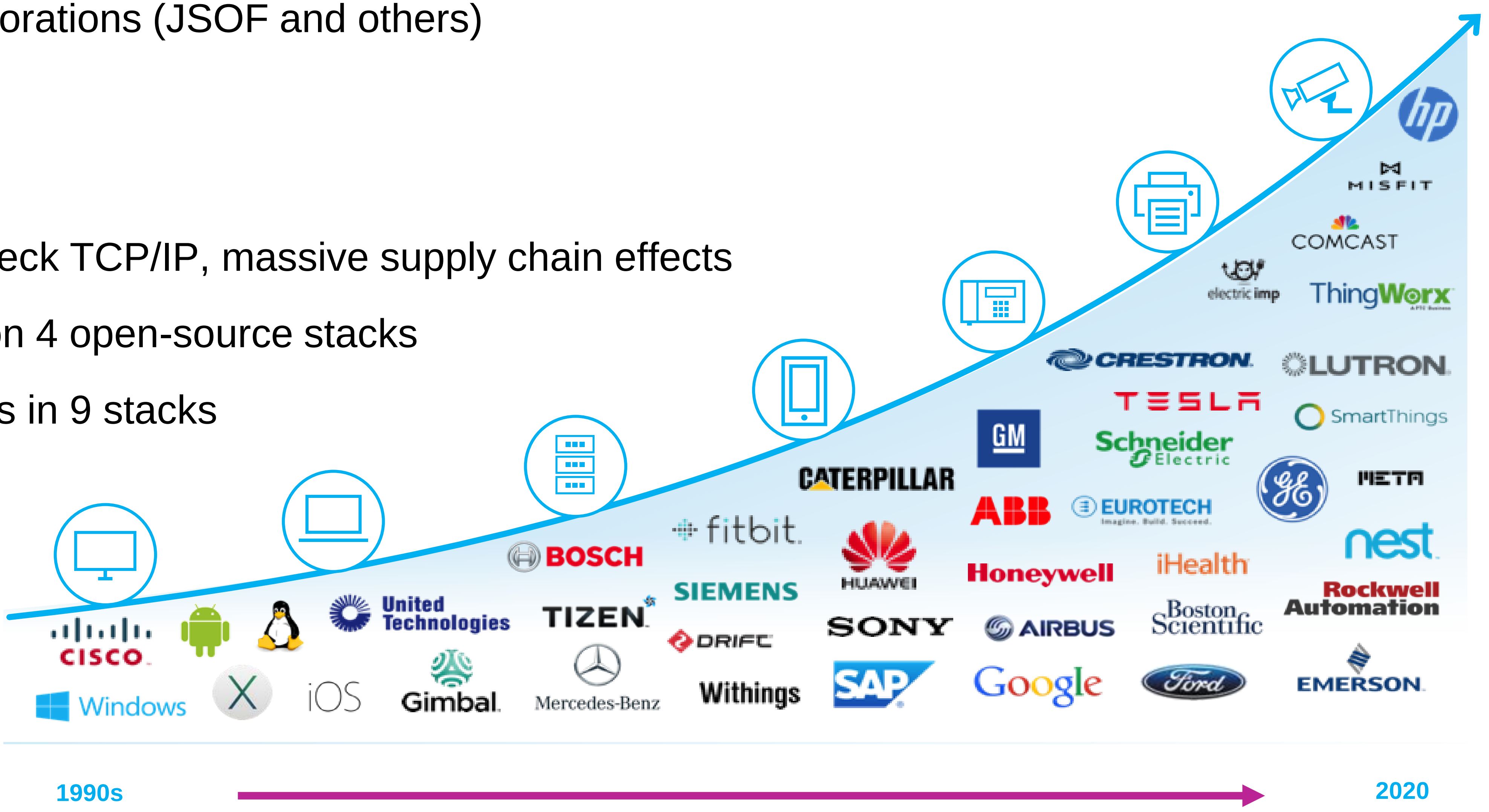
Project Memoria

Goal: large study of embedded TCP/IP stack security

- Why are they vulnerable? How are they vulnerable? What to do about it?
- Forescout Research Labs + collaborations (JSOF and others)

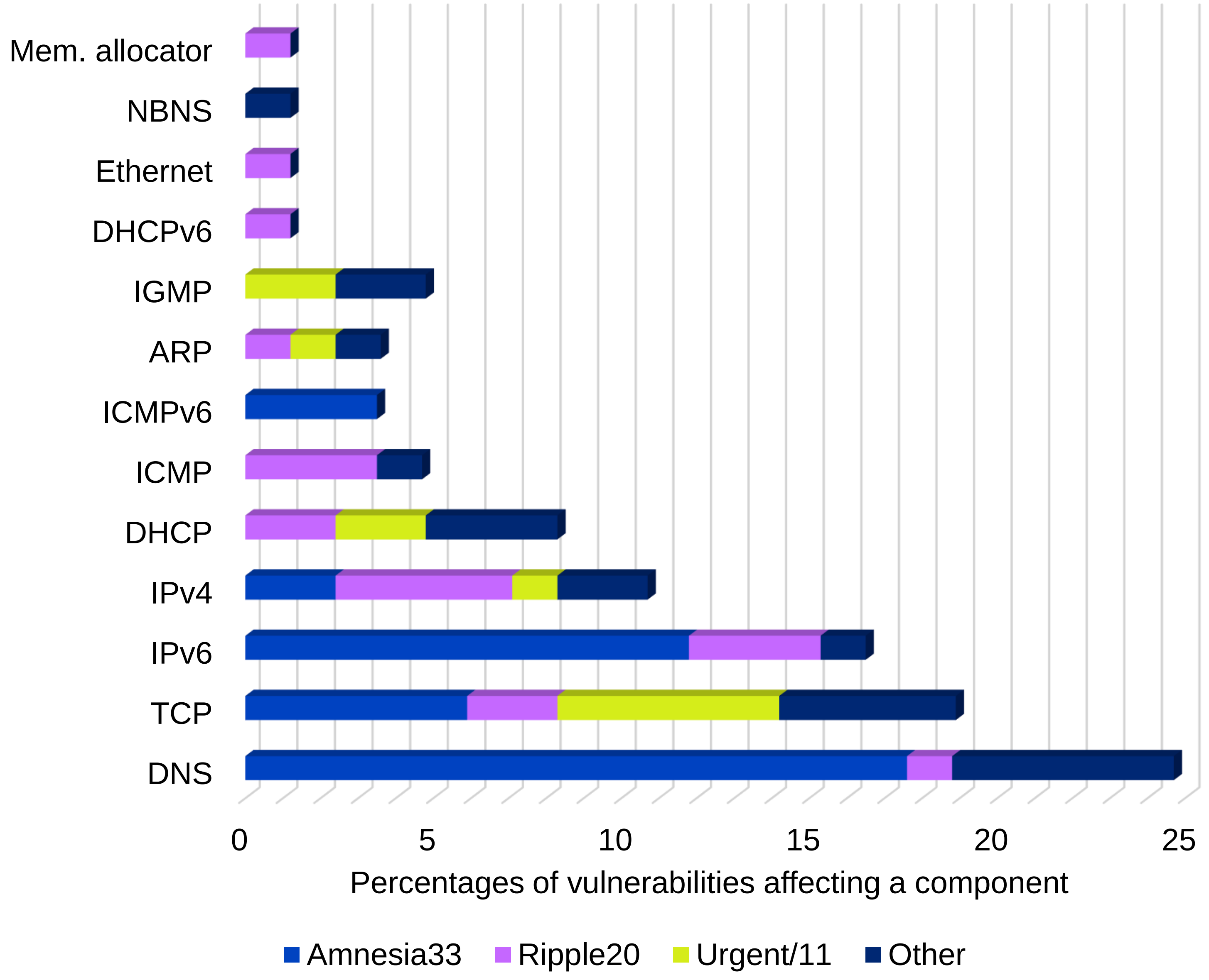
Previous research

- Ripple20 – 19 vulnerabilities on Treck TCP/IP, massive supply chain effects
- AMNESIA:33 – 33 vulnerabilities on 4 open-source stacks
- NUMBER:JACK – predictable ISNs in 9 stacks



https://www.forescout.com/research-labs/project-memoria

Why look more closely at DNS?



DNS is the most affected TCP/IP component in previous research

- Ripple20 – CVE-2020-11901 RCE
- AMNESIA:33 – 15 CVEs on DNS clients, 3 RCEs

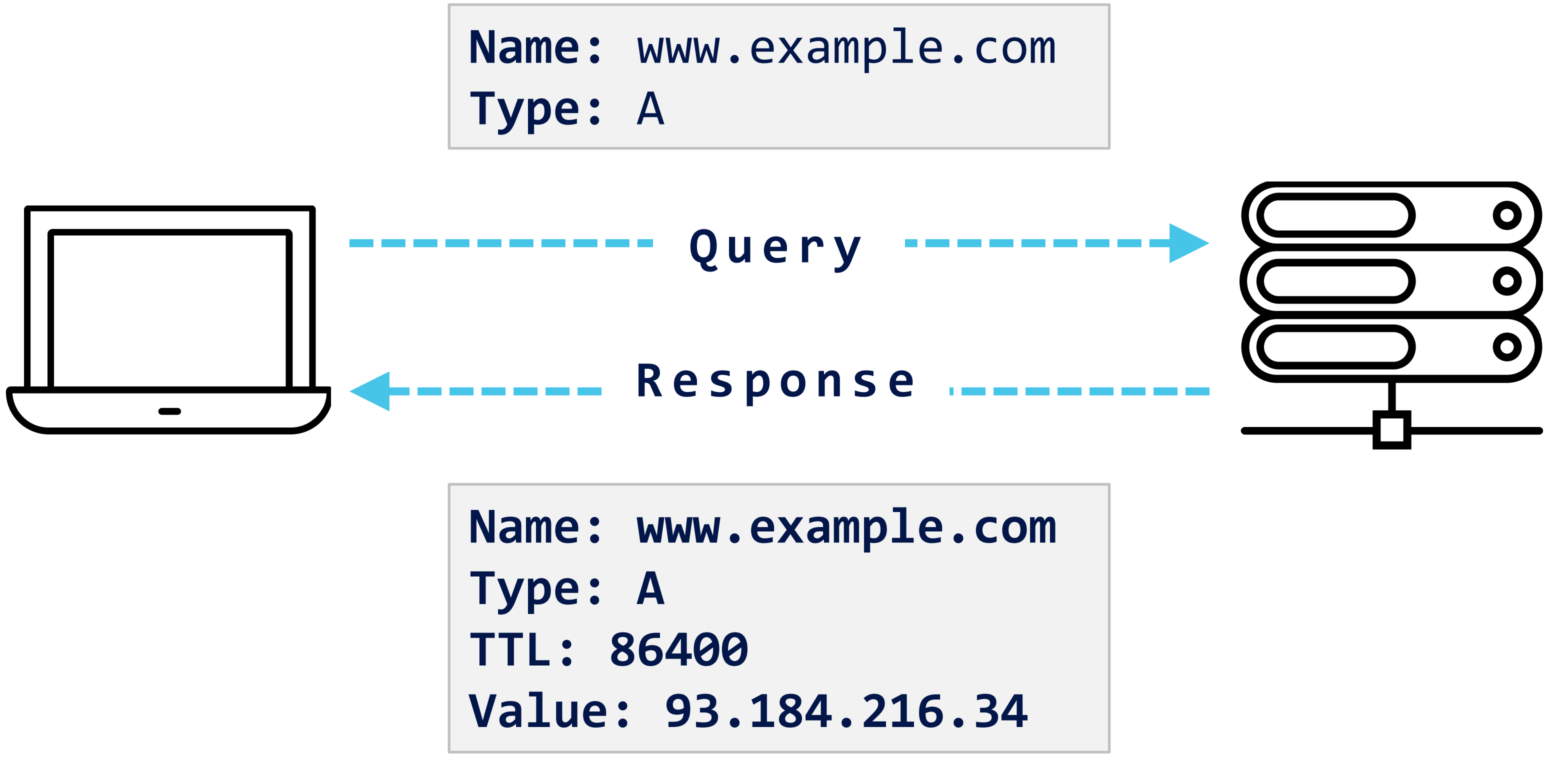
Protocol complexity is a good predictor of vulnerabilities – other major findings

- DNSpooq – 7 CVEs on dnsmasq
- SIGRed, SAD DNS, …

Typically externally accessible – large attack surface

Domain Name System (DNS)

- ○ Map between domain names and IP addresses
- ○ Client resolves name by querying DNS server
- ○ DNS server looks up the name and returns a response

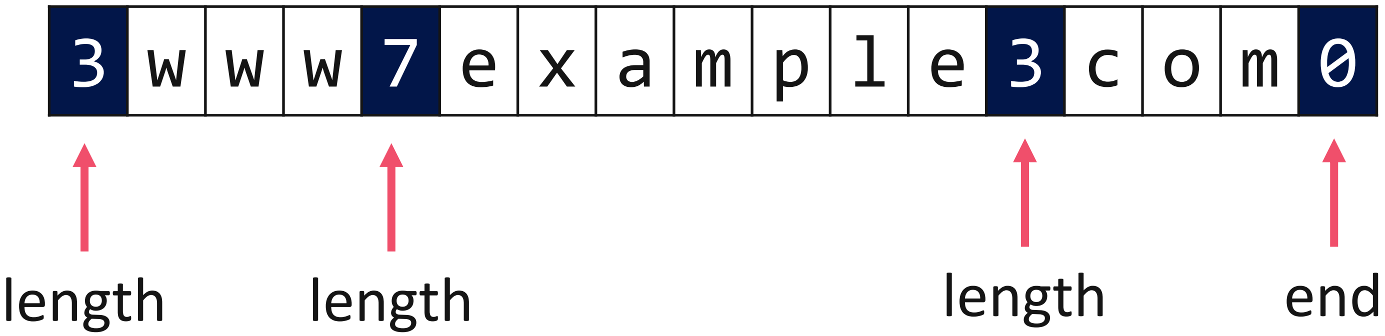


```
Name: www.example.com
Type: A
```

```
Name: www.example.com
Type: A
TTL: 86400
Value: 93.184.216.34
```

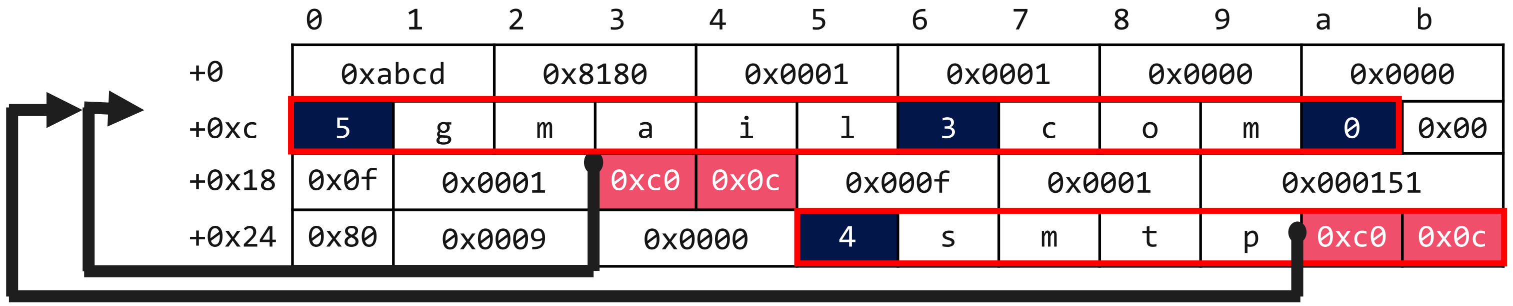
DNS encoding and compression

- Domain names are **sequences of labels**
- Each label **preceded by length byte**
- **Compression** replaces sequence of labels with **pointer to prior occurrence** of the same sequence
- Pointer encoded in two bytes: *0b11 + offset*
- Message compression is **also used** in [DHCP](), [mDNS](), [IPv6 Router Advertisement]()



3	w	w	w	7	e	x	a	m	p	l	e	3	c	o	m	0

length length length end



	0	1	2	3	4	5	6	7	8	9	a	b
+0	0xabcd		0x8180		0x0001		0x0001		0x0000		0x0000	
+0xc	5	g	m	a	i	l	3	c	o	m	0	0x00
+0x18	0x0f	0x0001		0xc0	0x0c	0x000f		0x0001		0x000151		
+0x24	0x80	0x0009		0x0000		4	s	m	t	p	0xc0	0x0c

11	offset

0 2 16

20 years of compression vulnerabilities

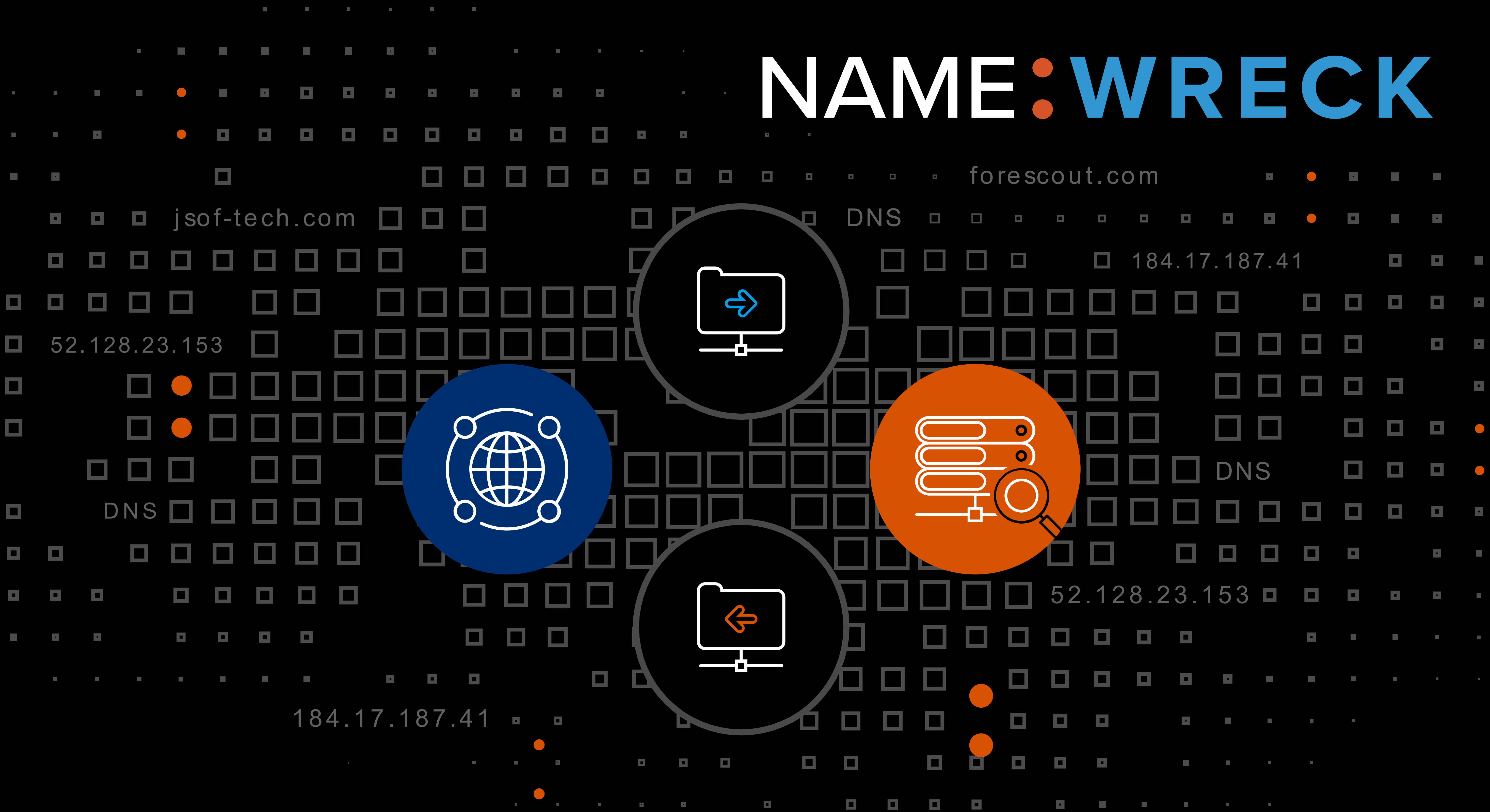
> *One problem with DNS compression is the **amount of code required to parse it.** Reliably locating all these names takes quite a bit of work that would otherwise have been unnecessary for a DNS cache. LZ77 compression would have been much easier to implement.*

– D.J. Bernstein, 2001
https://cr.yp.to/djbdns/notes.html

#	Vulnerability	Affected Products	Year
1	CVE-2000-0333	tcpdump, ethereal	2000
2	CVE-2002-0163	Squid	2002
3	CVE-2004-0445	Symantec DNS client	2004
4	CVE-2005-0036	Cisco IP Phone+	2005
5	CVE-2006-6870	Avahi	2006
6	CVE-2011-0520	MaraDNS	2011
7	CVE-2017-2909	Mongoose	2017
8	CVE-2018-20994	TrustDNS	2018
9	CVE-2020-6071	VLC	2020
10	CVE-2020-6072	VLC	2020
11	CVE-2020-12663	Unbound	2020
12	CVE-2020-11901	Treck TCP/IP stack (Ripple20)	2020
13	CVE-2020-24335	uIP TCP/IP stack (AMNESIA:33)	2020
14	CVE-2020-24339	PicoTCP TCP/IP stack (AMNESIA:33)	2020

+others: Windows Server 2008 R2 SP1 (2013, no CVE)

NAME:WRECK


forescout.com
jsof-tech.com
DNS
184.17.187.41
52.128.23.153
DNS
DNS
52.128.23.153
184.17.187.41

NAME:WRECK

Goal

- Analyze the DNS message **compression** feature in **several TCP/IP stacks**

What we quickly saw

- Good **potential for RCEs**
- No support for compression seems like a good way to avoid additional bugs

Research	Stack	Remark
Ripple20	Treck TCP/IP	Vulnerable (RCE)
AMNESIA:33	picoTCP	Vulnerable
	uIP	Vulnerable
	Nut/Net	Compression not supported Other DNS vulnerabilities
	lwIP	Compression not supported
	cycloneTCP	Not vulnerable
	uC/TCP-IP	Not vulnerable

First results

Selected stacks

- Typical **IT**, popular **embedded**, and new **IoT**
- Mix of **open-source** and **proprietary**
- Oldest from 90s (e.g., FreeBSD and Nucleus NET), newest from 2015 (Zephyr)

Stack	Vendor	Version analyzed
FreeBSD	Open-source	12.1
FreeRTOS+TCP	Open-source	2.2.2
IPnet	Wind River	VxWorks 6.6
NetX	Micrososft	6.0.1
nRF5 SDK	Nordic	15.2.0
Nucleus NET	Siemens	4.3
OpenThread	Open-source	20191113
Zephyr	Open-source	2.3.0

First results

- FreeRTOS+TCP, OpenThread and Zephyr **not vulnerable**
- **nRF5** SDK has **two out-of-bounds** reads but Nordic said it's experimental code → no CVE (discussion in the impact section)

New vulnerabilities

General observations

- FreeBSD: vulnerable **DHCP** client
- IPnet: bug collision, discovered by Exodus and fixed by Wind River in 2016. **No CVE at the time**
- NetX: reported as DoS because of Microsoft's response. We believe it might be a difficult to exploit RCE

Nucleus NET: looked for one type of vulnerability, but found several following Anti-Patterns

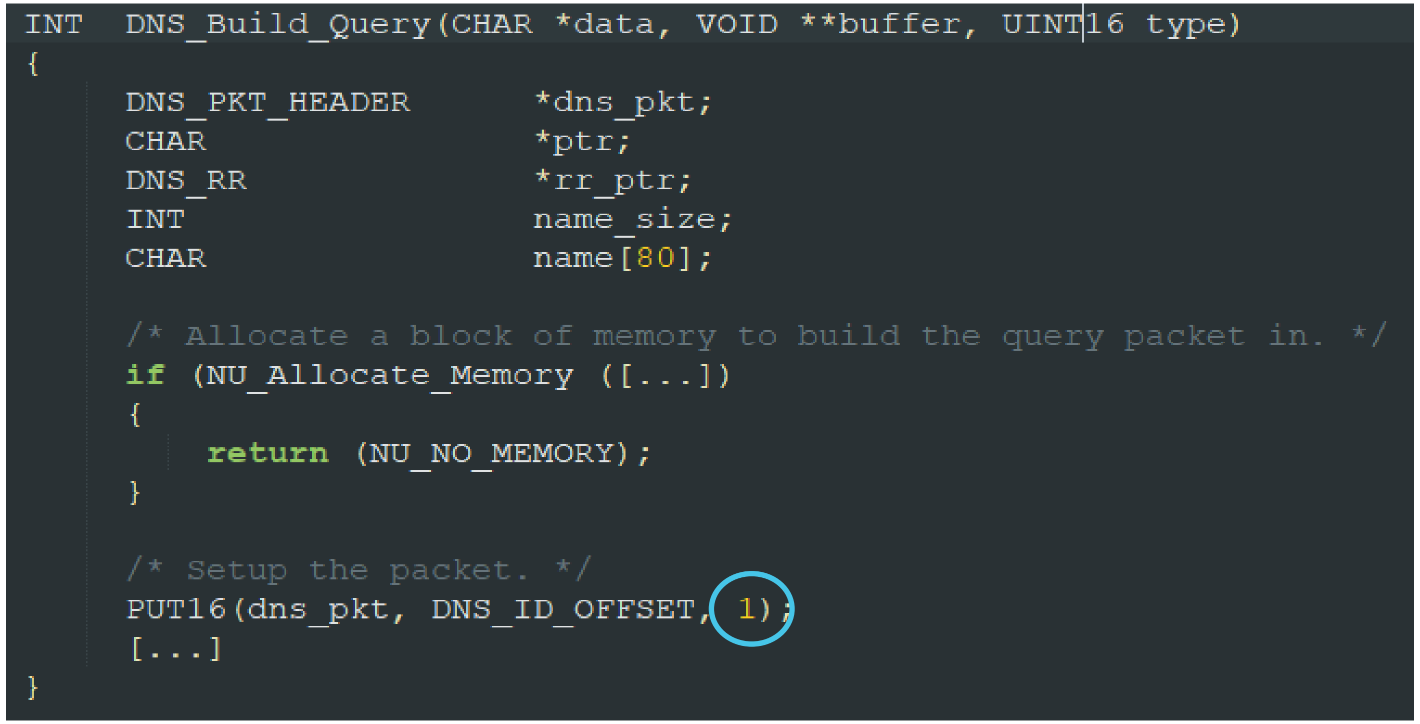
- Detailed discussion in the next slides

#	CVE	Stack	Feature	Potential Impact
1	CVE-2020-7461	FreeBSD 12.1	Message compression (DHCP client)	RCE
2	CVE-2016-20009	IPnet (VxWorks 6.6)	Message compression	RCE
3	CVE-2020-15795	Nucleus NET 4.3	Domain name label parsing	RCE
4	CVE-2020-27009	Nucleus NET 4.3	Message compression	RCE
5	CVE-2020-27736	Nucleus NET 4.3	Domain name label parsing	DoS
6	CVE-2020-27737	Nucleus NET 4.3	Domain name label parsing	DoS
7	CVE-2020-27738	Nucleus NET 4.3	Message compression	DoS
8	CVE-2021-25677	Nucleus NET 4.3	Transaction ID	DNS cache poisoning
9	*	NetX 6.0.1	Message compression	DoS

Anti-Pattern #1

Lack of TXID validation, insufficiently random TXID and source UDP port

- Source UDP port and Transaction ID (TXID) used by DNS clients/servers to match queries/responses
- Both must be difficult to predict, otherwise attackers can spoof DNS replies that will be accepted by a vulnerable client



```
INT  DNS_Build_Query(CHAR *data, VOID **buffer, UINT16 type)
{
    DNS_PKT_HEADER       *dns_pkt;
    CHAR                 *ptr;
    DNS_RR               *rr_ptr;
    INT                  name_size;
    CHAR                 name[80];

    /* Allocate a block of memory to build the query packet in. */
    if (NU_Allocate_Memory ([...])
    {
        return (NU_NO_MEMORY);
    }

    /* Setup the packet. */
    PUT16(dns_pkt, DNS_ID_OFFSET, 1);
    [...]
}
```

CVE-2021-25667 in Nucleus NET 4.3

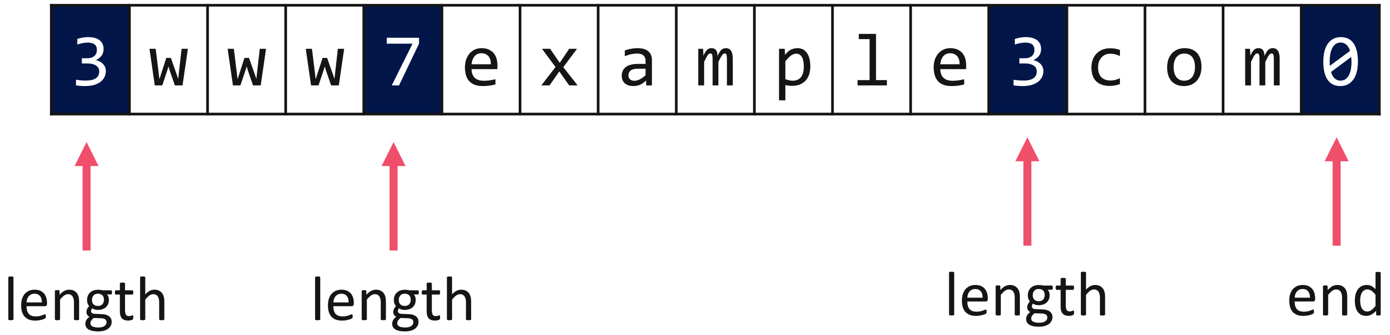
Issues observed:

- TXID of replies not validated (CVE-2020-17439 in uIP)
- TXID of requests set to constant (CVE-2020-17470 in FNET)
- **CVE-2021-25667** combines both: TXID is a constant which is not used for matching. Plus, the source UDP port value is predictable (same generator as TCP ISN)

Anti-Pattern #2

Lack of labels and name length validation

- Domain name labels should be <= 63 chars
- Domain names should be <= 255 chars
- Lengths should be **validated** according to data in packet



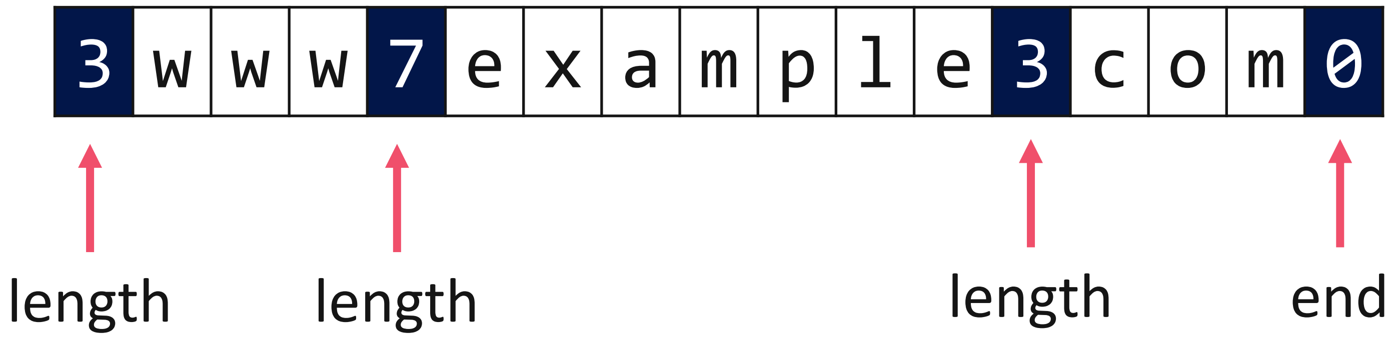
Issues observed:

- No restriction on lengths, allowing attackers to craft longer payloads
- Length values copied directly from network packet and used for the size of heap or stack buffers. Absence of bounds checks then allows attackers to control the allocation of these buffers
- **CVE-2020-15795** in Nucleus NET: no check whether the reported lengths match the number of bytes in a domain name

Anti-Pattern #3

Lack of NULL-termination validation

- Domain names must end with a NULL byte (0x00)
- Implementations should not just assume, but validate it
- Attacker-controlled placement of NULL byte in a domain name + lax domain name and label length checks may result in controlled memory reads and writes



Issues observed:

- Even when the domain name boundary checks are implemented, absence of checks for NULL byte leads to memory-related off-by-one errors, causing Denial-of-Service
- **CVE-2020-27736 in Nucleus NET**

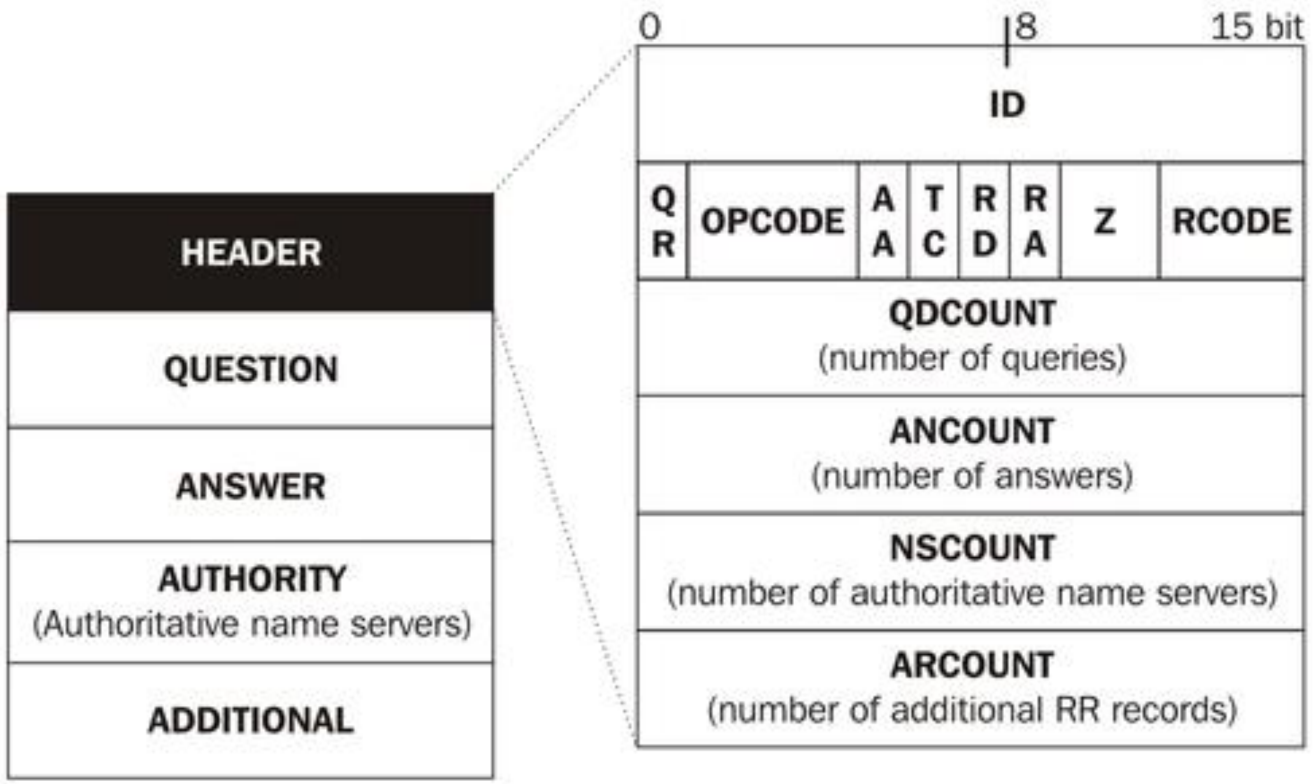
Anti-Pattern #4

Lack of record count fields validation

- DNS header contains four count fields for records
- After the header comes the data of individual records
- Packets with incorrect QCOUNT/ANCOUNT/NSCOUNT/ARCOUNT values should be dropped (RFC5625)

Issues observed:

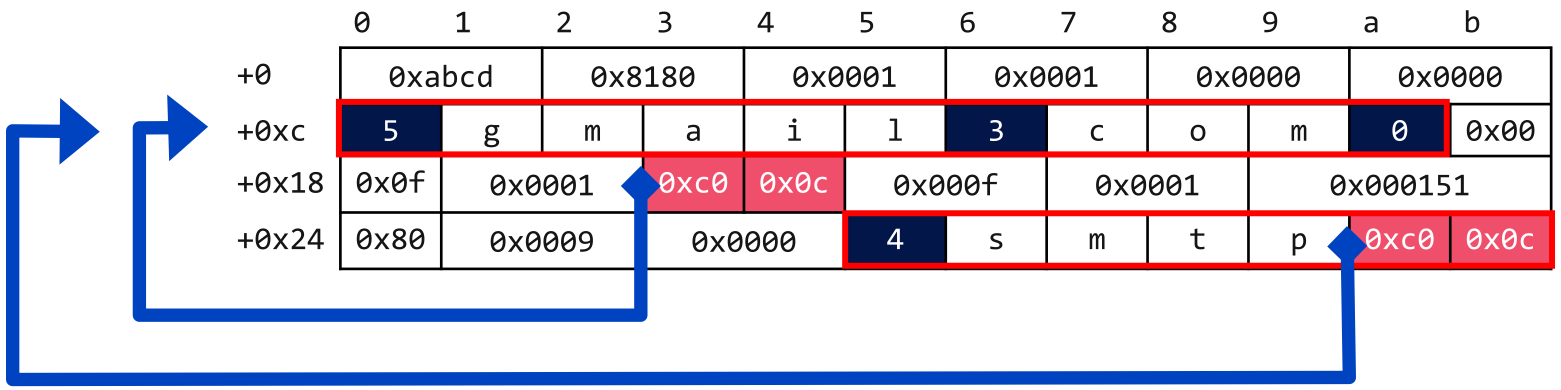
- Record count fields taken from the packet but no validation whether the packet has enough data to hold the specified numbers of records
- CVE-2020-27737 in Nucleus NET: by providing fewer answers than set in ANCOUNT, attackers may cause a Denial-of-Service when the code reads out of bounds of the packet as it tries to parse answer records that do not exist



Anti-Pattern #5

Lack of domain name compression pointer and offset validation

- Code must check that compression offset in incoming packet points “backwards” and lands on a valid uncompressed domain name
- Otherwise, it is possible to craft offset values pointing “forward”, allowing the attackers to “hijack” the DNS parser
- The same compression pointer should not be followed more than once



	0	1	2	3	4	5	6	7	8	9	a	b
+0	0xabcd		0x8180		0x0001		0x0001		0x0000		0x0000	
+0xc	5	g	m	a	i	l	3	c	o	m	0	0x00
+0x18	0x0f	0x0001		0xc0	0x0c	0x000f		0x0001		0x000151		
+0x24	0x80	0x0009		0x0000		4	s	m	t	p	0xc0	0x0c

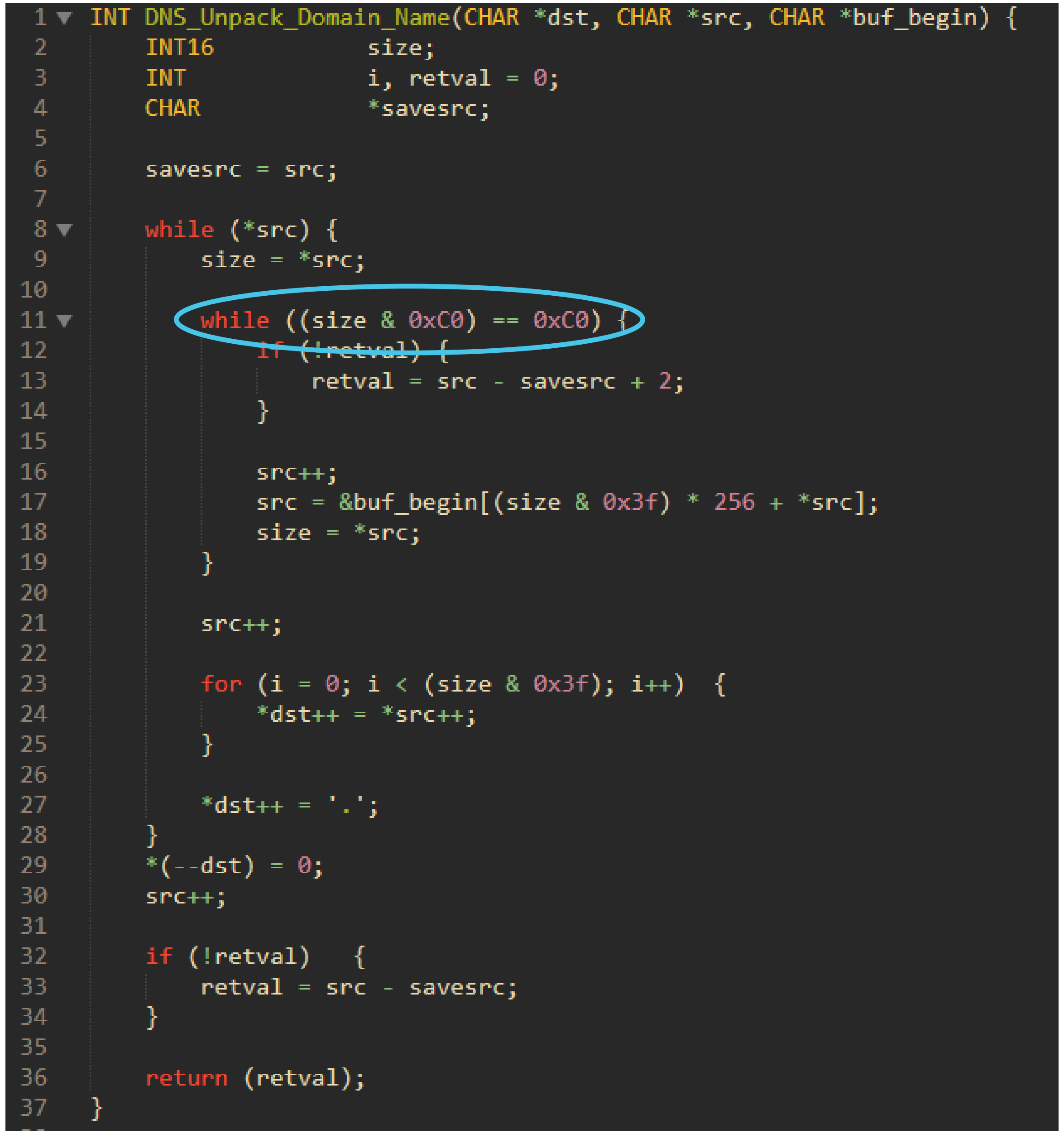
Issues observed:

- Value of compression pointer often unchecked. Since it is a 14-bit value, it can point to 16383 (0x3fff) bytes past the beginning of the DNS header. If the packet is shorter than this value the code might read out of bounds
- If the pointer points to itself, it might cause the parsing code to enter an infinite loop
- Not checking or mis-calculating the decompressed name length
- **CVEs on FreeBSD, IPnet, Nucleus NET, NetX**

Exploiting a message compression bug

Usually a combination of individual issues (example with Nucleus NET):

- CVE-2020-27009: attacker can craft a DNS response packet with a combination of invalid compression pointer offsets that allows them to write arbitrary data
- CVE-2020-15795: attacker can craft meaningful code to be injected by abusing very large domain name records in the malicious packet
- CVE-2021-25667: attacker can bypass DNS query-response matching to deliver the malicious packet to the target



```c
INT DNS_Unpack_Domain_Name(CHAR *dst, CHAR *src, CHAR *buf_begin) {
    INT16           size;
    INT             i, retval = 0;
    CHAR            *savesrc;

    savesrc = src;

    while (*src) {
        size = *src;

        while ((size & 0xC0) == 0xC0) {
            if (!retval) {
                retval = src - savesrc + 2;
            }

            src++;
            src = &buf_begin[(size & 0x3f) * 256 + *src];
            size = *src;
        }

        src++;

        for (i = 0; i < (size & 0x3f); i++)  {
            *dst++ = *src++;
        }

        *dst++ = '.';
    }
    *(--dst) = 0;
    src++;

    if (!retval)   {
        retval = src - savesrc;
    }

    return (retval);
}
```

CVE-2020-27009

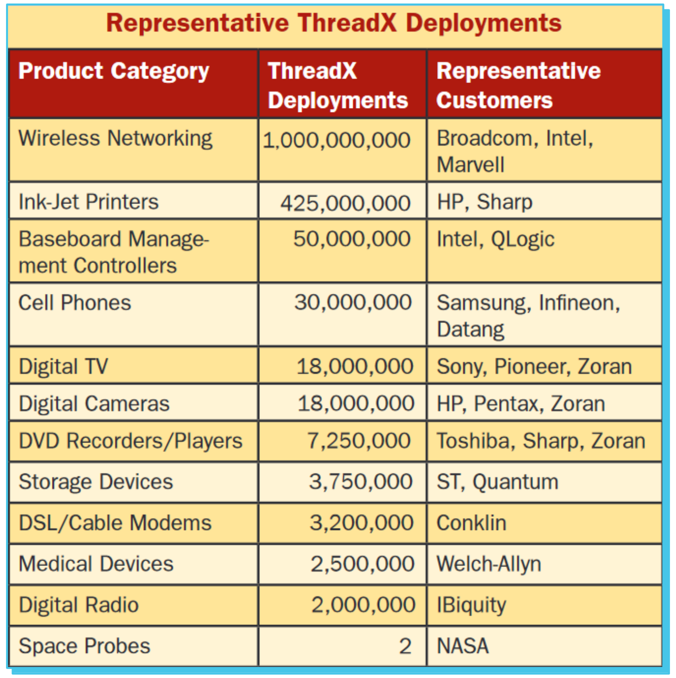
Details on the new report +
https://www.youtube.com/watch?v=wo_YhLBVkrY
(Ripple20)

Impact

Affected devices

Understading affected vendors/devices is difficult because TCP/IP stacks are reused multiple times in many ways (see Ripple20 & AMNESIA:33)

- FreeBSD is very popular in web and storage servers, but also is the basis of several **popular appliances and other software** (https://en.wikipedia.org/wiki/List_of_products_based_on_FreeBSD)
- Nucleus RTOS (Nucleus NET), ThreadX (NetX), VxWorks (IPnet) used for **decades in critical systems**
- Altogether, more than 10 billion deployments. Not all OS deployments use “default” stack, not all have DNS/DHCP client enabled and not every version is vulnerable. But **1% of 10 billion is 100 million...**



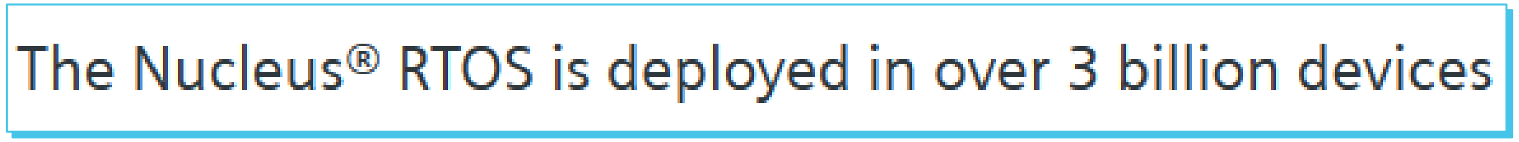
Representative ThreadX Deployments

Product Category	ThreadX Deployments	Representative Customers
Wireless Networking	1,000,000,000	Broadcom, Intel, Marvell
Ink-Jet Printers	425,000,000	HP, Sharp
Baseboard Management Controllers	50,000,000	Intel, QLogic
Cell Phones	30,000,000	Samsung, Infineon, Datang
Digital TV	18,000,000	Sony, Pioneer, Zoran
Digital Cameras	18,000,000	HP, Pentax, Zoran
DVD Recorders/Players	7,250,000	Toshiba, Sharp, Zoran
Storage Devices	3,750,000	ST, Quantum
DSL/Cable Modems	3,200,000	Conklin
Medical Devices	2,500,000	Welch-Allyn
Digital Radio	2,000,000	IBiquity
Space Probes	2	NASA

https://www.pertech.co.il/wp-content/uploads/2016/03/el_brochure_2012.pdf

The World's Leading Companies Trust VxWorks

https://www.windriver.com/products/vxworks#customers



The Nucleus® RTOS is deployed in over 3 billion devices

https://www.mentor.com/embedded-software/nucleus

Again, the supply chain

NAME:WRECK

another example of vulnerabilities that trickle down the supply chain because of popular components, which makes vulnerability management hard

Illustrative issue 1: IPnet/VxWorks 6.6

- Vulnerability from 2016 that was silently patched (CVE-2016-20009). Fixed in at least some devices (e.g., Huawei firewalls), but which?
- Affects currently unsupported versions of VxWorks, but several examples of currently supported devices running VxWorks 5 from 20 years ago (e.g., Dell PowerConnect IT switches and Siemens SCALANCE ICS switches). We have not checked if these are vulnerable, there could be patches via extended support.

Illustrative issue 2: Nordic nRF5 SDK

- Vendor mentioned vulnerability is not in production software, but “experimental code” in SDK. However, developers tend to use this type of code from SDKs in production devices.
- See “Leveraging Flawed Tutorials for SeedingLarge-Scale Web Vulnerability Discovery” and “An Empirical Study of C++ Vulnerabilities in Crowd-Sourced Code Examples”

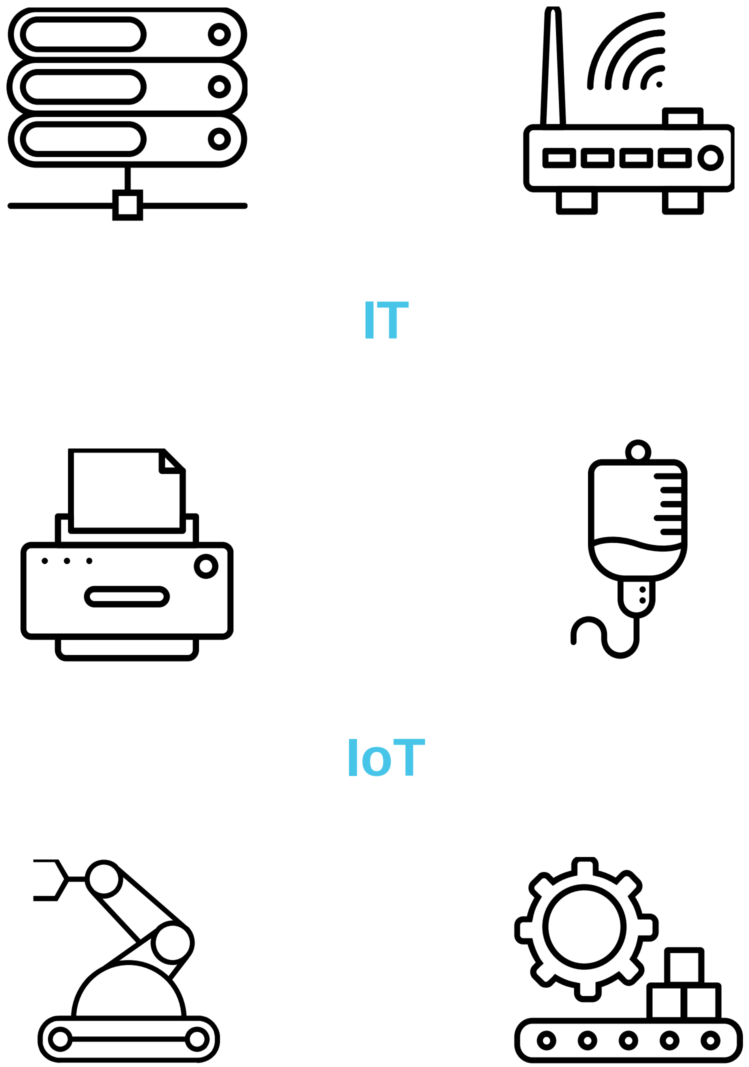
Different types of impact

Exploitation

- FreeBSD is a modern OS with exploit mitigation and sandboxing
- The others typically run on constrained hardware with barely any memory protection

Patching

- FreeBSD: often IT servers that are easy to identify and patch centrally (SSH, high availability, etc)
- The others run on very specific firmware and mission-critical devices



IT

IoT

OT

Mitigation

Developers

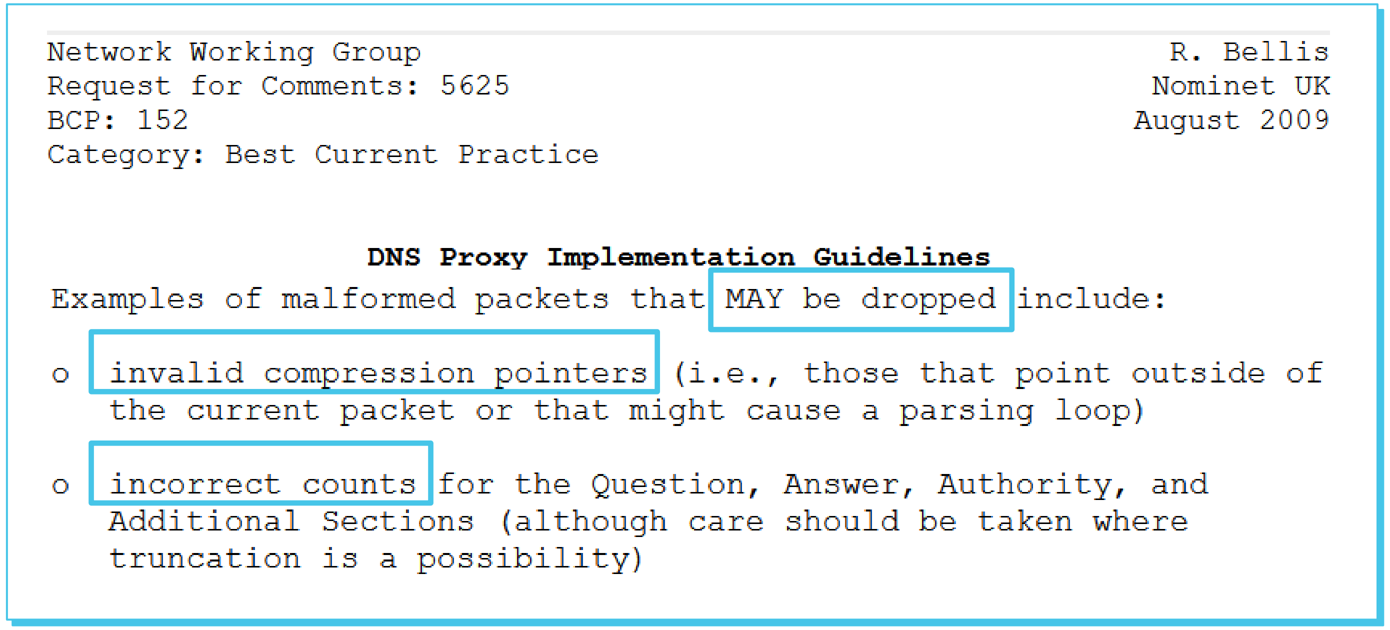
- Better documentation
- Static analysis

Network operators

- Device fingerprinting
- Intrusion detection

Better documentation

- Specification and security information is **scattered across RFCs**, which are often complex, ambiguous, or outdated. This and previous research shows the drastic security effects of this situation



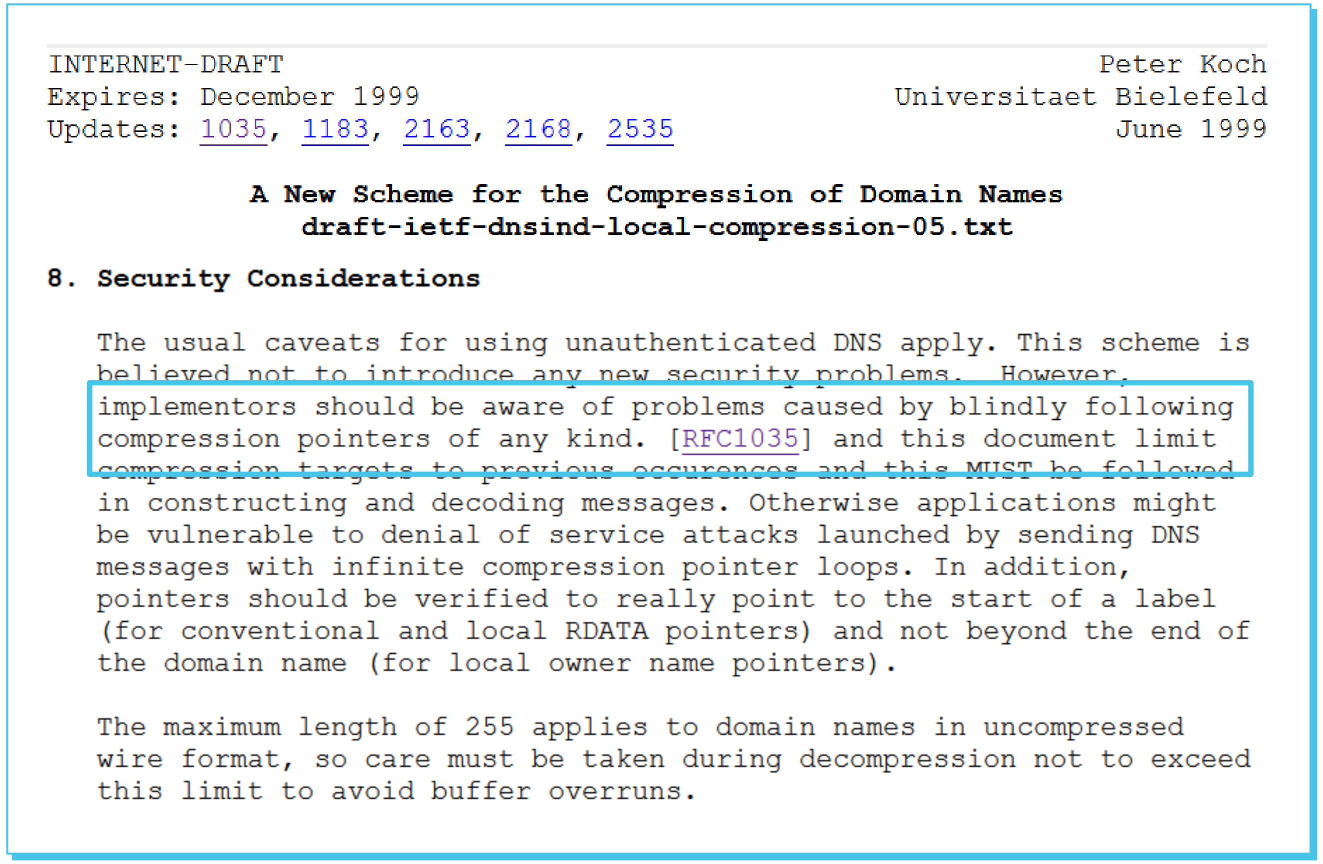
Network Working Group R. Bellis
Request for Comments: 5625 Nominet UK
BCP: 152 August 2009
Category: Best Current Practice

DNS Proxy Implementation Guidelines

Examples of malformed packets that MAY be dropped include:

o invalid compression pointers (i.e., those that point outside of the current packet or that might cause a parsing loop)

o incorrect counts for the Question, Answer, Authority, and Additional Sections (although care should be taken where truncation is a possibility)



INTERNET-DRAFT Peter Koch
Expires: December 1999 Universitaet Bielefeld
Updates: 1035, 1183, 2163, 2168, 2535 June 1999

A New Scheme for the Compression of Domain Names
draft-ietf-dnsind-local-compression-05.txt

8. Security Considerations

The usual caveats for using unauthenticated DNS apply. This scheme is believed not to introduce any new security problems. However, implementors should be aware of problems caused by blindly following compression pointers of any kind. [RFC1035] and this document limit compression targets to previous occurences and this MUST be followed in constructing and decoding messages. Otherwise applications might be vulnerable to denial of service attacks launched by sending DNS messages with infinite compression pointer loops. In addition, pointers should be verified to really point to the start of a label (for conventional and local RDATA pointers) and not beyond the end of the domain name (for local owner name pointers).

The maximum length of 255 applies to domain names in uncompressed wire format, so care must be taken during decompression not to exceed this limit to avoid buffer overruns.

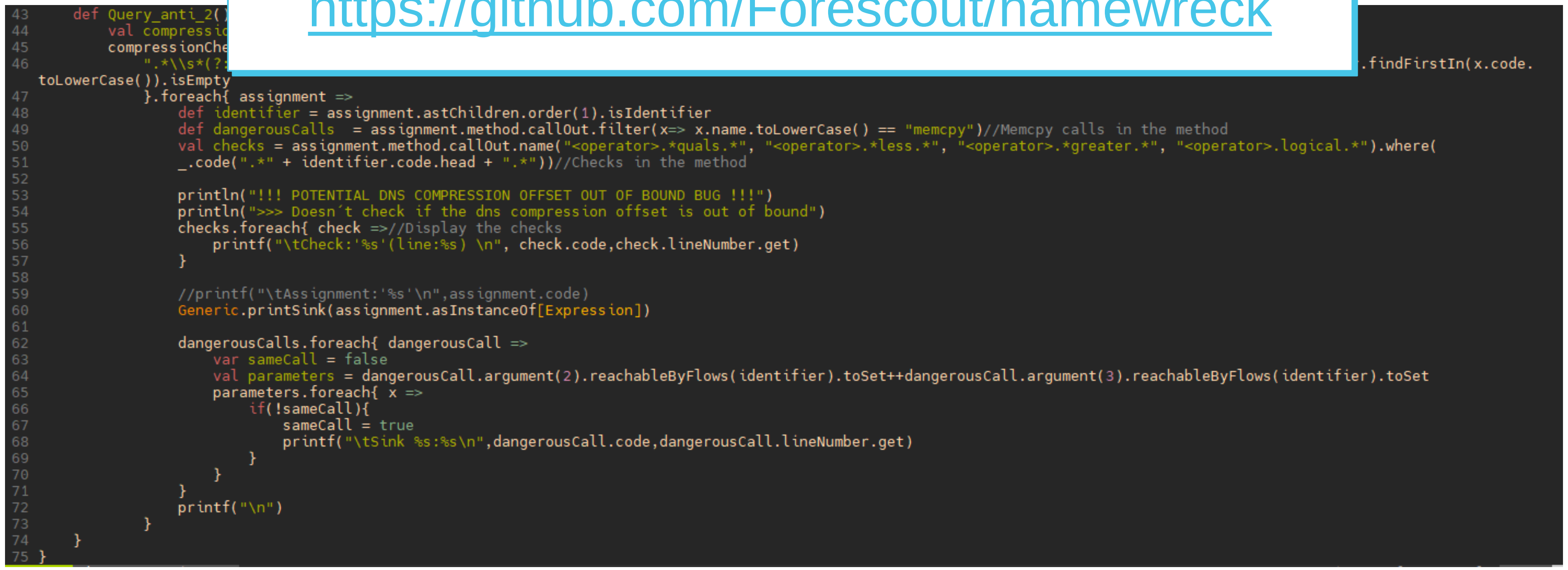
- We wrote an informational RFC draft about the identified anti-patterns and how to avoid them

Static analysis

- Developers need tools to **readily spot potential bugs**
- We created code to **identify some anti-patterns using Joern**, an open-source code querying tool for C/C++

https://joern.io/

https://github.com/Forescout/namewreck

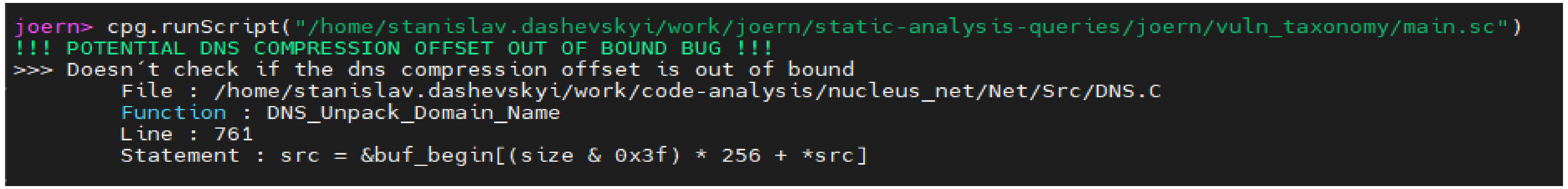


```
    def Query_anti_2()
        val compressio
        compressionChe
            ".*\\s*(?:                                                                  .findFirstIn(x.code.
toLowerCase()).isEmpty
            }.foreach{ assignment =>
                def identifier = assignment.astChildren.order(1).isIdentifier
                def dangerousCalls  = assignment.method.callOut.filter(x=> x.name.toLowerCase() == "memcpy")//Memcpy calls in the method
                val checks = assignment.method.callOut.name("<operator>.*quals.*", "<operator>.*less.*", "<operator>.*greater.*", "<operator>.logical.*").where(
                _.code(".*" + identifier.code.head + ".*"))//Checks in the method

                println("!!! POTENTIAL DNS COMPRESSION OFFSET OUT OF BOUND BUG !!!")
                println(">>> Doesn´t check if the dns compression offset is out of bound")
                checks.foreach{ check =>//Display the checks
                    printf("\tCheck:'%s'(line:%s) \n", check.code,check.lineNumber.get)
                }

                //printf("\tAssignment:'%s'\n",assignment.code)
                Generic.printSink(assignment.asInstanceOf[Expression])

                dangerousCalls.foreach{ dangerousCall =>
                    var sameCall = false
                    val parameters = dangerousCall.argument(2).reachableByFlows(identifier).toSet++dangerousCall.argument(3).reachableByFlows(identifier).toSet
                    parameters.foreach{ x =>
                        if(!sameCall){
                            sameCall = true
                            printf("\tSink %s:%s\n",dangerousCall.code,dangerousCall.lineNumber.get)
                        }
                    }
                }
                printf("\n")
            }
    }
}
```

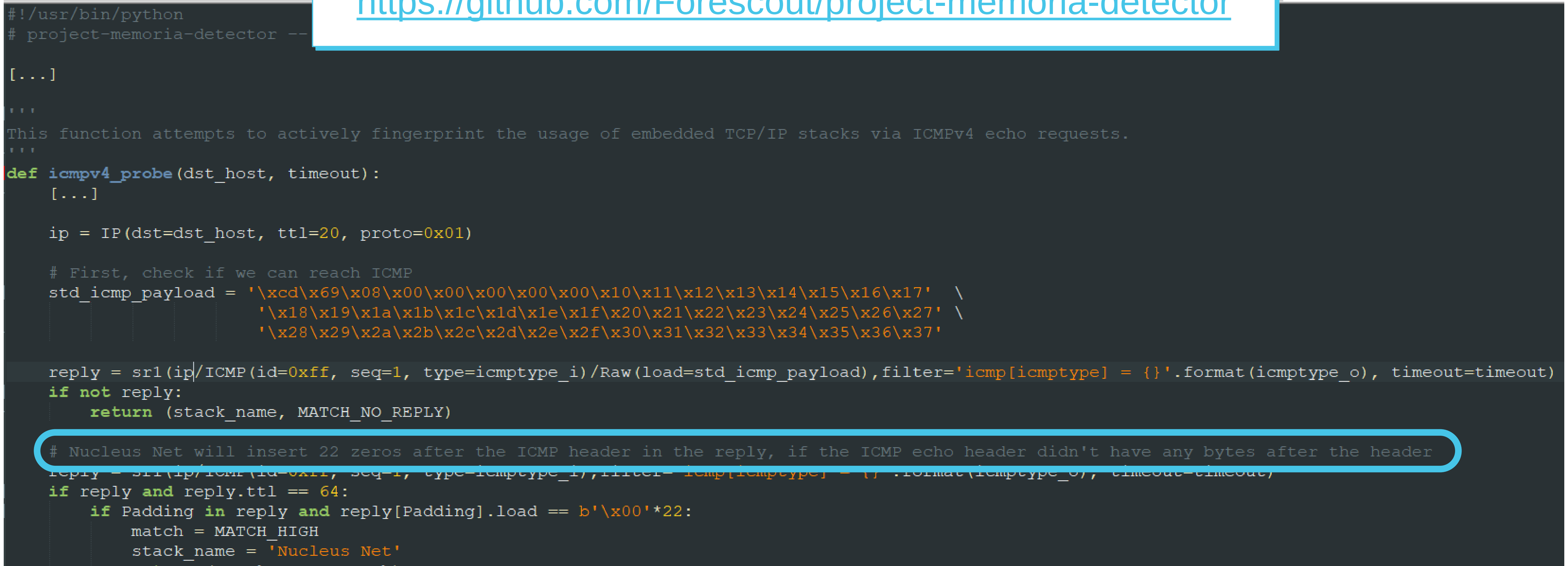


```
joern> cpg.runScript("/home/stanislav.dashevskyi/work/joern/static-analysis-queries/joern/vuln_taxonomy/main.sc")
!!! POTENTIAL DNS COMPRESSION OFFSET OUT OF BOUND BUG !!!
>>> Doesn´t check if the dns compression offset is out of bound
        File : /home/stanislav.dashevskyi/work/code-analysis/nucleus_net/Net/Src/DNS.C
        Function : DNS_Unpack_Domain_Name
        Line : 761
        Statement : src = &buf_begin[(size & 0x3f) * 256 + *src]
```

Device fingerprinting

- ○ Embedded stacks typically have implementation quirks, often useful for stack fingerprinting
- ○ ICMP replies and TCP options are a prime example
- ○ Accurate fingerprinting enables other mitigations – patching and segmentation

https://github.com/Forescout/project-memoria-detector



```
#!/usr/bin/python
# project-memoria-detector --

[...]

'''
This function attempts to actively fingerprint the usage of embedded TCP/IP stacks via ICMPv4 echo requests.
'''
def icmpv4_probe(dst_host, timeout):
    [...]

    ip = IP(dst=dst_host, ttl=20, proto=0x01)

    # First, check if we can reach ICMP
    std_icmp_payload = '\xcd\x69\x08\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17'  \
                        '\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x22\x23\x24\x25\x26\x27' \
                        '\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37'

    reply = sr1(ip/ICMP(id=0xff, seq=1, type=icmptype_i)/Raw(load=std_icmp_payload),filter='icmp[icmptype] = {}'.format(icmptype_o), timeout=timeout)
    if not reply:
        return (stack_name, MATCH_NO_REPLY)

    # Nucleus Net will insert 22 zeros after the ICMP header in the reply, if the ICMP echo header didn't have any bytes after the header
    reply = sr1(ip/ICMP(id=0xff, seq=1, type=icmptype_i),filter='icmp[icmptype] = {}'.format(icmptype_o), timeout=timeout)
    if reply and reply.ttl == 64:
        if Padding in reply and reply[Padding].load == b'\x00'*22:
            match = MATCH_HIGH
            stack_name = 'Nucleus Net'
```

Exploit detection rules

PACKETS NOT CONFORMING TO THE FOLLOWING RULES SHOULD BE DROPPED OR THEIR PRESENCE ALERTED

Invalid domain label, name, and resource data lengths

- Domain label length must be 0>n>64
- Number of domain label characters must correspond to the value of the domain label byte
- Domain name length must be <= 255 bytes
- NULL terminator must be present at the end of domain name
- Value of data length byte (RDLENGTH) must reflect the number of bytes available in the field that describes the resource (RDATA)

Invalid compression pointers

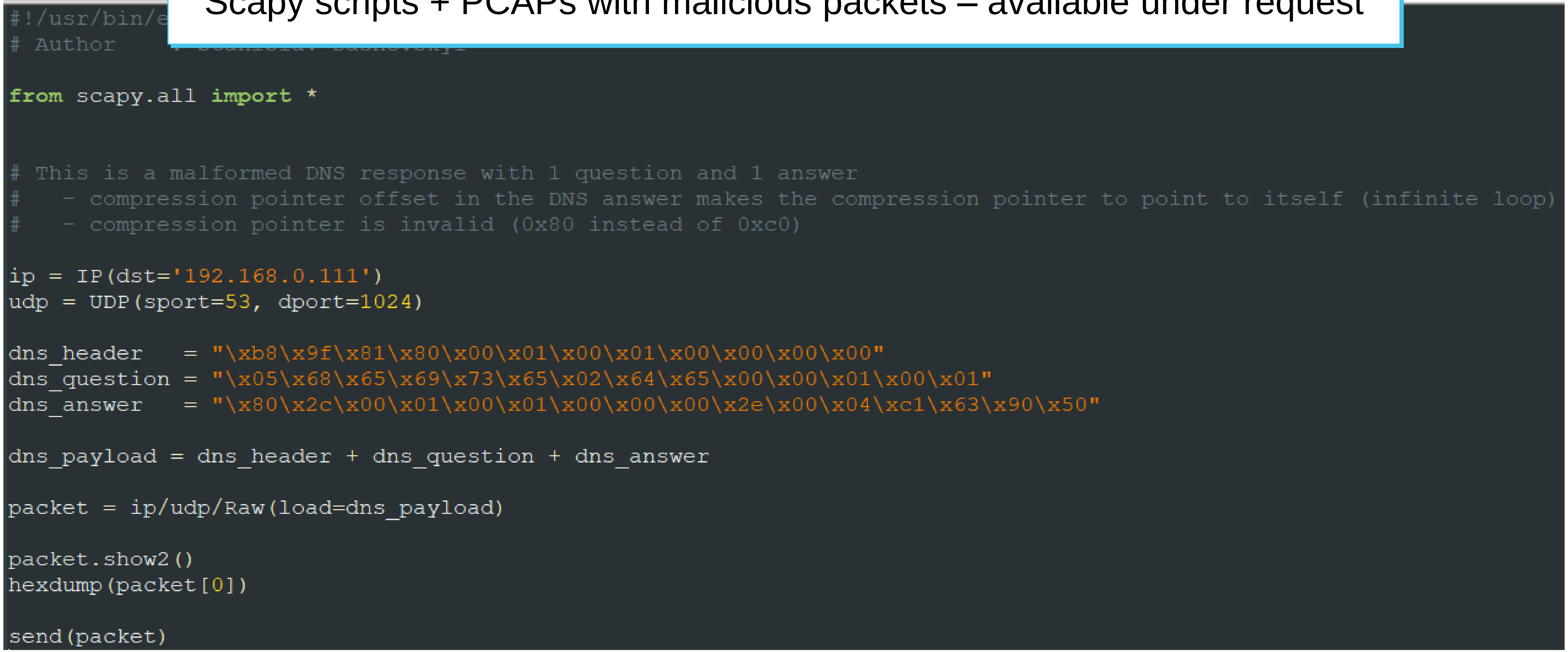
- Compression pointer must resolve to a byte within a DNS record with a value 0>n>64
- Offset of this byte must be < offset of the compression pointer
- Compression pointers must not be “followed” more than once

Invalid record counts

- Values of the header count bytes (QCOUNT/ANCOUNT/NSCOUNT/ARCOUNT) must correspond to the actual data present within the packet

Intrusion detection

Scapy scripts + PCAPs with malicious packets – available under request



```
#!/usr/bin/e
#  Author

from scapy.all import *


# This is a malformed DNS response with 1 question and 1 answer
#   - compression pointer offset in the DNS answer makes the compression pointer to point to itself (infinite loop)
#   - compression pointer is invalid (0x80 instead of 0xc0)

ip = IP(dst='192.168.0.111')
udp = UDP(sport=53, dport=1024)

dns_header   = "\xb8\x9f\x81\x80\x00\x01\x00\x01\x00\x00\x00\x00"
dns_question = "\x05\x68\x65\x69\x73\x65\x02\x64\x65\x00\x00\x01\x00\x01"
dns_answer   = "\x80\x2c\x00\x01\x00\x01\x00\x00\x00\x2e\x00\x04\xc1\x63\x90\x50"

dns_payload = dns_header + dns_question + dns_answer

packet = ip/udp/Raw(load=dns_payload)

packet.show2()
hexdump(packet[0])

send(packet)
```

Conclusion

Conclusions

RFC mis-implementation is a common cause of vulnerabilities in TCP/IP stacks

- RFCs are sometimes complex, ambiguous, or outdated
- DNS clients have several vulnerabilities, but **message compression stands out: very common and often RCE**

Not implementing support for compression is an effective mitigation against this type of vulnerability

- Since the bandwidth saving associated to this type of compression is almost meaningless in a world of fast connectivity, DNS message compression currently seems to introduce more problems than it solves

DNS clients seem to be tested less rigorously than servers for security

- Because clients communicate with a limited set of servers (instead of a large set of clients), they may be prone to vulnerabilities being detected later in the development cycle and potentially remaining for longer in production software
- Not only for TCP/IP stacks, every DNS implementation should be tested: firewalls, IDS, packet dissectors, forwarders, etc.

Key takeaways

DNS complexity leads to critical vulnerabilities

- 50% of what we analyzed is vulnerable to a specific anti-pattern
- That means many other implementations are probably vulnerable

Popular TCP/IP stacks amplify the problem

- Vulnerable code runs in millions of devices

There are several steps to mitigate this problem

- Report about vulnerabilities & anti-patterns: https://www.forescout.com/research-labs/namewreck
- Draft Informational RFC & Open-source Joern queries: https://github.com/Forescout/namewreck
- Open-source fingerprinting of stacks: https://github.com/Forescout/project-memoria-detector
- Malicious PCAPs: research@forescout.com

Thank you!

FORESCOUT®

Active Defense for the Enterprise of Things™

JSOF

✉ research@forescout.com

www.forescout.com/research-labs

in www.linkedin.com/company/forescout-technologies

www.twitter.com/forescout

✉ shlomi@jsof-tech.com

www.jsof-tech.com

in www.linkedin.com/company/jsof

www.twitter.com/jsof18