Off-Path Hacking: The Illusion of Challenge-Response Authentication

Mrs.Sowbhagya M P
Asst. Professor, Dept. of ISE,
City Engineering College,
Bangalore
Visvesvaraya Technological
University, Karnataka, INDIA
sowbhagya.mp@gmail.com

Ms. Tejaswini
Student, Department of ISE,
City Engineering College,
Bangalore
Visvesvaraya Technological
University, Karnataka, INDIA
tejir15@gmail.com

Ms. Priyanka
Student, Department of ISE,
City Engineering College,
Bangalore
Visvesvaraya Technological University,
Karnataka, INDIA
priyanka.nataraj@yahoo.com

Abstract—Everyone is concerned about Internet security, yet most traffic is not cryptographically protected. Typical justification is that most attackers are off-path and cannot intercept traffic; hence, intuitively, challenge-response defenses should suffice to ensure authenticity. Of- ten, the challenges re-use existing header fields to protect widely- deployed protocols such as TCP and DNS. We argue that this practice may often give an illusion of security. We review recent off-path TCP injection and DNS poisoning attacks, enabling attackers to circumvent existing challenge-response defenses. Both TCP and DNS attacks are non-trivial, yet practical. The attacks foil widely deployed security mechanisms, and allow a wide range of exploits, such as long-term caching of malicious objects and scripts. We hope that this review article will help improve defenses against off-path attackers. In particular, we hope to motivate, when feasible, adoption of cryptographic mechanisms such as SSL/TLS, IPsec and DNSSEC, providing security even against stronger Man-in-the-Middle attackers.

Keywords: off-path attacks, DNS cache poisoning, TCP injections, challenge-response defenses

I. INTRODUCTION

Since 1989 [1], experts have been arguing that Internet security requires cryptographic protocols, ensuring security against Man-in-the-Middle (MitM) attackers. A MitM attacker is located on the path of the communicating parties, and can manipulate the communication between them in any way, i.e., intercept, modify, block and inject spoofed packets; see the Monster in the Middle in Figure 1.

The information security community invested significant efforts in developing cryptographic schemes and protocols, standards and products, providing security against MitM attackers, such as IPsec, SSL/TLS, Secure- BGP and DNSSEC. In spite of all these efforts, and al- though Internet security is well recognised to be critical, most Internet traffic is still not cryptographically protected. For example, we found that only about 6% of the TCP traffic is cryptographically protected with SSL/TLS (based on CAIDA dataset of 3 million packets [2]); and less than 1% of the DNS resolvers enforce DNSSEC (cryptographic) validation [3].

We believe that the main reason for the under utilisation of cryptography, is an illusion of security againstnetwork-based attacks, due to two false beliefs. The first false belief is that in reality, attackers can rarely obtain MitM capabilities, and even when they can, they are reluctant to do so since such activities may lead to detection. We claim that this is incorrect; there are common scenarios where attackers may obtain MitM capabilities, e.g., by accessing wireless communication, by manipulations of the largely unprotected routing mechanisms, or by controlling some intermediate device. Furthermore, such attacks are often carried out, without detection and repercussions, e.g., route hijacking occurs frequently [4].

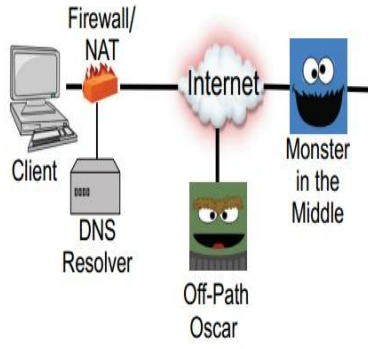

Fig. 1. Off-Path Attacker Network Model.

However, in this review, we focus on the second false belief, which is that current, non-cryptographic, Internet protocols already provide sufficient protection against typical, common attackers, and in particular, against off- path attackers.

Unlike a MitM attacker, an off-path attacker cannot observe or modify legitimate packets sent between other parties; however, he can transmit packets with a spoofed (fake) source IP address - impersonating some legitimate party, as illustrated byOff-Path Oscar in Figure 1. Spoofed packets are used in many attacks, most notably, in Denial of Service (DoS) attacks. Significant efforts are made to make spoofing less readily available to attackers, most notably ingress filtering ([RFC3704]). However, IP spoofing is still possible via many ISPs; hence, the IP-spoofing ability is often available.

Our main goal in this review is to convince that this second belief is (also) false, and that current Internet protocols are often vulnerable even to an off-path attacker. Specifically, we discuss a few recent results that allow off-path attacks on basic Internet protocols: traffic injection into TCP connections and DNS cache poisoning.

The key to the off-path attacks that we discuss is circumvention of challenge-response defenses. Challenge- response defenses are often relied upon to distinguish between (spoofed) packets from an off-path attacker and (legitimate) packets from legitimate communication end- point. In order to authenticate a response from a server, a client sends a random challenge with the request, which is echoed in the response. Since an off-path attacker, which we dub Oscar, cannot eavesdrop on packets exchanged between the server and the client, it appears that Oscar would have to guess the challenge; hence, the (sufficiently long, random) challenge allows to prevent Oscar from crafting a packet with a valid response.

The security of most Internet applications, e.g., email, web surfing, and most peer-to-peer applications, relies on challenge-response mechanisms, mainly as part of the underlying TCP and DNS protocols. For example, the widely-used web-security mechanisms based on cookies and other 'same origin policy' mechanisms, depend on the security of both TCP and DNS.

We review vulnerabilities, allowing off-path attacks on both TCP and DNS in (common) scenarios, i.e., where Oscar circumvents the existing challenge-response mechanisms. Challenge-response defenses may fail in several ways:

Insufficient entropy: challenges may be insufficiently-long or non-uniform. Both types were abused in attacks against old implementations of DNS [8] and TCP [9].

Piggybacking: attacker may 'piggyback' fake con- tent, onto valid responses (containing correct challenges), exploiting IP fragmentation. Such attacks were presented for DNS [10] and TCP [11], [12].

Side-channels: attacker may reduce the entropy of the challenge, by exploiting the fact that challenges mostly or wholly reuse existing protocol fields. Namely, challenges are fields which already exist in requests and are echoed in responses for some other purpose. Such attacks were presented for DNS [11], [12], [13] and TCP [7], [14], [15], [16], [17].

The root cause of many of these attacks is the at- tempt to retrofit security, and in this case incorporate a challenge-response mechanism, into an existing protocol. By reusing existing protocol fields, the defenses were deployed only by changing the clients, and without coordinated changes in servers (and the protocol itself). Such defenses are much easier to deploy - but also easier to attack. Specifically, we discuss attacks that allow an off-path attacker to learn the 'dual-use' challenge fields. This allows off-path TCP injection and DNS cache poisoning.

II. History of Off-Path Attacks

TCP and DNS are basic protocols, and off-path attacks on their authenticity - TCP injection and DNS poisoning - impact almost all Internet applications. As such, it is a common belief that they ensure integrity against off-path attackers. However, security against off-path (or MitM) attackers was not of the original design goals of these protocols, and only minimal changes were done to thespecifications to support challenge-response defenses, e.g., selecting identifiers at random.

In Figure 2 we present a 'time-line' of important attacks on both TCP (upper row) and DNS (lower row).

The time-line begins in 1985, with publication of a TCP injection attack based on the use of predictable sequence numbers [7], and Bellovin's seminal paper from 1989 [1], pointing out that security should not be based on the presumed off-path protection of DNS and TCP. Bellovin presented vulnerabilities of (some) TCP implementations to off-

path attacks, and discussed potential exploits and defenses.

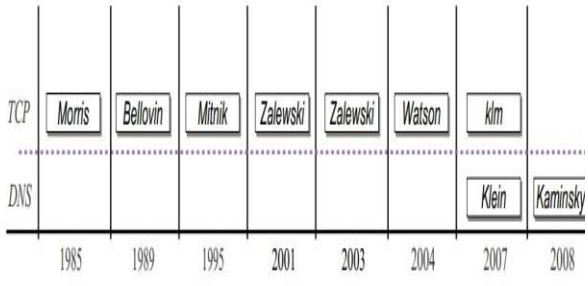

Fig. 2. Time-line of DNS Poisoning and TCP Injection attacks.

Unfortunately, in spite of these warnings, until 1995 most TCP stacks still used trivially-predictable initial sequence numbers (ISN). This changed only after the notorious TCP injection attack by Mitnick on Shimomura [14]. After the attack, many implementations changed to 'less predictable' ISN choices. However,in2001, Zalewski showed that most implementations are still 'sufficiently predictable', allowing off-path attacks; this motivated adoption of more random choice of ISNs in most operating systems, as standardized in [RFC6528].

In 2003, Zalewski also commented that 'piggybacking' on fragmented TCP traffic may allow injection at- tacks [13]; the piggybacking attack was improved in [12], and exploited for DNS poisoning in [10].

A special TCP injection attack was presented by Watson in 2004. This attack only injected a 'RST' packet, breaking up a connection, and focused on long-lived connections using known client (and server) ports and addresses, as used at the time by the Internet routing protocol BGP. To address this concern, many TCP implementations also began using 'unpredictable' client ports.

In 2007, there were two surprising results: (1) a TCP injection attack presented by the pseudonym author klm in Phrack magazine [17], and (2) a DNS poisoning attack exploiting poor random-number generators. Both attacks were clever and significant, although with limited scope. In particular, the attack on TCP worked only against Windows machines, connected directly to the Internet (rather than via firewall, as usually is the case), and did not handle concurrent connections.

Kaminsky presented an even more significant DNS poisoning attack in 2008, which allowed efficient off-path poisoning of most DNS resolver implementations at the time [8] (see Section 2). The response to this attack was rapid adoption of additional 'patches', mostly, more challenge-response fields, increasing the length of the random challenge and therefore (hopefully) making the attack impractical; the most notable patch was source port randomisation (SPR); see [RFC5452].

III. Malicious Agents

Some of the off-path attacks require, in addition to the spoofing ability, also a malicious agent in the victim's network or host. We briefly explain the different agent models.

A zombie is a machine controlled by the adversary, e.g., compromised by malware, in the victim's network.

A puppet is a weaker agent: a restricted malicious script or applet running in web-browser sandbox. Attacks relying on a puppet agent require (only) that a client in the victim network 'surfs' to the attacker's web-site, enabling the adversary to run such a script. The script is restricted by same origin policy (described in [RFC6454]), and can only communicate via the browser, i.e., request (and receive) HTTP objects (no access to TCP/IP packet headers).

IV. DNS CACHE POISONING

The Domain Name System (DNS) provides name to address mapping for services in the Internet. DNS name servers maintain the mappings for services in the do- mains for which they are authoritative, and DNS re- solvers are agents, used by clients, to retrieve the map- pings from the name servers. Resolvers send requests to name servers, and receive responses. Prior to accepting and caching the responses, they are validated; widely deployed validations rely on challenge-response mechanisms. The resolvers send the challenges, e.g., in form of a random 16-bit TXID field within the request, andvalidate that the same values appear in responses. We next explain that challenge-response defenses may fail even against weak, off-path attackers.

V. Kaminsky's DNS Cache Poisoning

In 2008, Kaminsky [9] presented an efficient cache poisoning attack against resolvers which authenticated responses using a random TXID and used a known (fixed) source port, which at that time was 53. The steps of the attack, illustrated in Figure 3, are the following:

(1) the attacker triggers a DNS request for a random sub-domain of the victim domain $1.foo.com.

(2) DNS resolver receives the request and forwards it to the target name server.

(3) the attacker then sends 216 responses with spoofed source IP (of the name server); each response is a referral mapping of the name server ns.foo.com to 6.6.6.6, an IP address controlled by the attacker.

(4) the response containing the correct TXID is accepted, cached and returned to the client.
(5) authentic DNS response is ignored, since there is no matching pending request. If the attack fails, i.e., an authentic response from the real name server arrived before the correct response from the attacker, the attack is repeated with a new random subdomain $2.foo.com
.

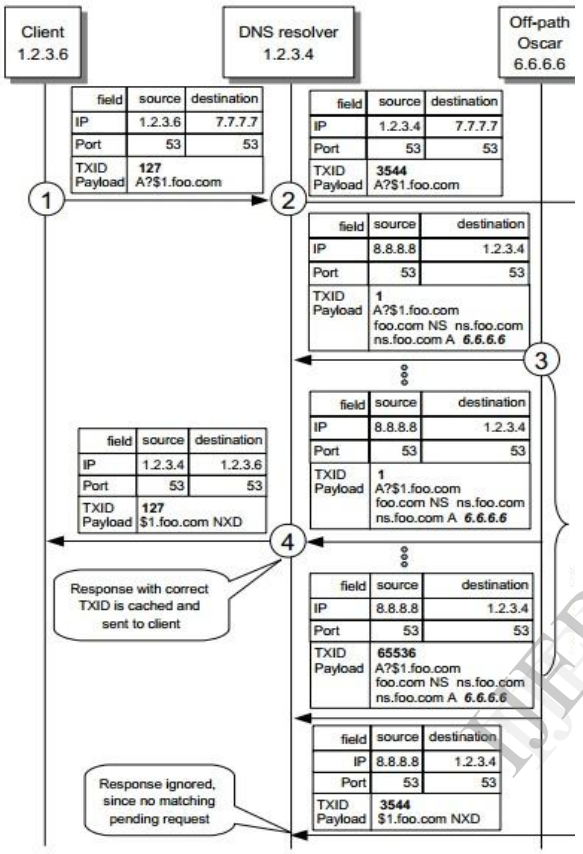

Fig. 3. Kaminsky's Attack.

VI. Vulnerability of Resolvers Behind NAT

Network Address Translation (NAT) devices are used to alleviate the problem of IPv4 addresses depletion in the Internet, by allocating non-unique addresses in local networks and sharing unique addresses between a number of internal hosts. The NAT devices modify the source ports in outbound packets in order to correlate between the inbound packets and the internal hosts.In this section we describe port derandomisation against per-destination ports allocation, implemented by many systems.

PER-DESTINATION PORTS ALLOCATION. For a tuple defined by (src-IP:src-port,dst-IP:dst-port,protocol),a per-destination NAT selects the first port at random, and subsequent ports are increased sequentially

PREDICT-THEN-POISON ATTACK. Off-path attacker, Oscar, controls a zombie, i.e., non-privileged malware that runs on a client host in the LAN. The attack is composed of two phases, illustrated in Figure 4: port prediction and poisoning.The port prediction phase, which allows bypass the SPR defense, works as follows:
(1) The zombie sends a packet to create a mapping in the NAT table; in the example in Figure 4 we assume arbitrarily that port 6666 was selected.
(2) Then, Oscar at address 6.6.6.6 sends 216 packets with a spoofed source IP of 8.8.8.8.
(3) The zombie increments this port by 1, in our example the result is 6667, and sends it to Oscar.

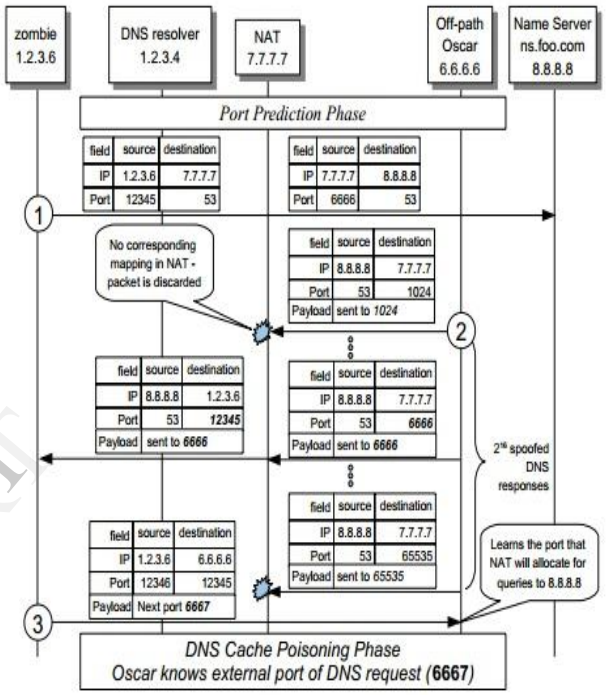

Fig. 4. Predict-then-Poison Attack, assuming per-destination NAT.

VII. TCP INJECTIONS

The Transmission Control Protocol (TCP) is the main transport protocol of the Internet, carrying most of the communication between clients and servers. The recent off-path TCP injection attacks operate in two phases: (1) Learn Connection Four-Tuple. Oscar, the off-path attacker, learns the four parameters of a TCP connection between a client and a server, that is, their respective IP addresses and ports. (2) Learn Sequence Number. Oscar learns the current sequence number, for packets sent from the server to the client or vice versa. After the attacker learns the connection four-tuple and one of the sequence numbers, he can inject data into the TCP connection, impersonating as one of the participating peers to the other. Table 1 surveys the techniques used in recent injection attacks for both phases and their requirements. In the reminder of this section we present a simple implementation for each phase.

	Learn Connection Four-Tuple	Learn Sequence
klm [19]	Remote probing for connection (Windows client, no firewall)	Side chann (Windows cli
Qian et al. [23], [22]	Local probing for connection, e.g., with netstat (Malware)	Read client system (Malware; in [23] also seq. #
Gilad and Herzberg [20]	Establish connection, exploit sequential port allocation impl. (Puppet, Windows client)	Side chann (Puppet, Window
Gilad and Herzberg [21]	Establish connection, client port derandomisation (Puppet, client behind firewall)	exploit HTTP-client st (Puppet, no TL

TABLE 1
Off-Path TCP Injection Attacks: Building Blocks. In parentheses:

In order to launch an injection attack, Oscar must first identify a TCP connection between the victim client and server. Some methods for identifying this connection scan the victim's machine, either remotely ([19]) or locally ([23], [22]); see Table 1.

In this subsection we describe a simple method which uses a puppet (script restricted by browser sandbox) running on the client machine to open such a connection. Since Oscar opens the connection he chooses the server, and the server's IP address and port are known. To find the client's IP address, the puppet sends a request to Oscar's site; this request contains the client's IP address.

The final challenge of this phase is to detect the client port. In [18] we showed how to break the randomised port selection algorithm which was standartised in [RFC6056], and used by Linux and Android clients; this attack exploits the TCP state machine, which leaks to the off-path attacker information about the choice of the client port.

SEQUENCE NUMBER LEARNING TECHNIQUE. The learning phase has two steps: Inject and Observe, illustrated in Figure 5. In the inject step, Oscar injects data into the stream of HTTP responses that the server sends to the client. This data is read in the observe step, which allows Oscar to determine the server's sequence number.

(A) Inject step. Let *wnd* denote the browser's receive- buffer for the connection and |*wnd*|denote its size. In order to inject the data, Oscar sends to the browser $\frac{2^{32}}{|wnd|}$ packets, spoofed to appear to be from the server (on its victim-connection with the client). The *i*th packet has server sequence number *i*-|*wnd*|; since the sequence number field is 32-bits long, exactly one of these packets has a 'valid' sequence number, which falls within the limits of wnd; all the other packets are discarded bythe client. Each of Oscar's packets contains as payload page(i) which is a simple web-page defined as follows:

```
<HTML><BODY>
<iframe src= "oscar.com/i.html" />
</BODY></HTML>
```

(B) Observe step. In this step, the puppet makes prevalent requests to the server, until it reaches the data injected by Oscar in the previous step. Each server- response that arrives at the client shifts wnd forward; after several such responses arrive, there is no gap of unreceived bytes between the injected data and the beginning of wnd. Then, the browser reads the injected- response, assuming that it corresponds to the request.

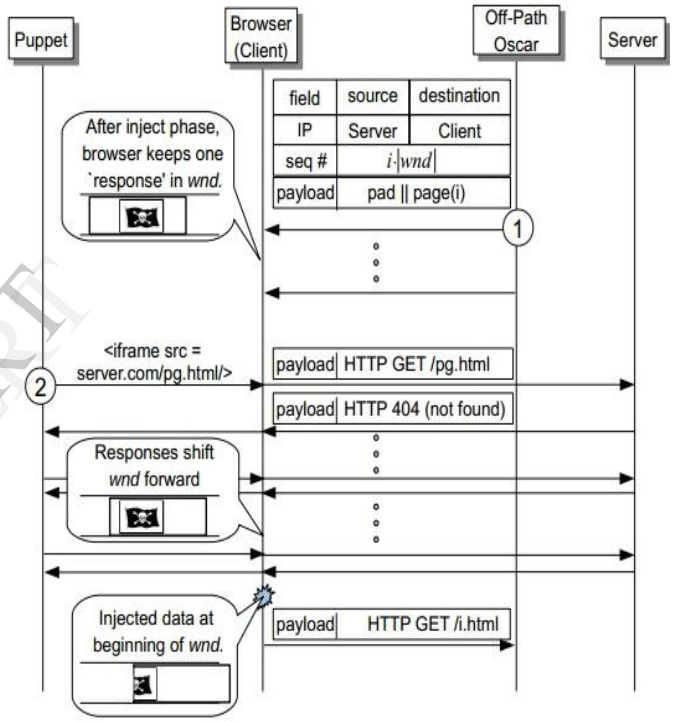

Fig. 5. Sequence Number Learning Technique.

VIII. EXPLOITING INJECTION AND POISONING

To conclude our discussion of off-path TCP injection and DNS poisoning attacks, we briefly discuss some potential exploits.

Exploiting DNS poisoning is straightforward. Both users and applications use DNS extensively to resolve domain names; DNS poisoning allows circumvention of security mechanisms, e.g., SPF and blacklists, and 'hijacking' of connection requests to legitimate servers. In particular, 'hijacking' can allow phishing, where a user thinks that he interacts with a trusted site, while he actually deals with a fake site (exposing credentials, in-stalling malware, etc.). The poisoned mapping is

cached and hence can impact many users of the resolver.

Exploiting TCP injections is more challenging, since TCP is a transport protocol and does not involve caching. However, in common scenarios, TCP injections can allow critical exploits. In particular, TCP injections suffice to circumvent the Same Origin Policy, hijack 'cookies' and cause execution of malicious scripts (XSS). In order to cause long-term impact similar to DNS poisoning,attackers can exploit caching of objects by web caches. By crafting the HTTP headers of his injected packets, Oscar can cache spoofed objects (e.g., web-pages) for long time.

IX. CONCLUSIONS

The techniques discussed in this article allow off-path attackers to circumvent main challenge-response defenses: source port randomisation and initial sequence number randomisation.

Our message is that defenses should be designed and analysed carefully, and not 'patched' by reusing existing fields whose entropy may be insufficient or reduced by side-channels. In particular, in order to prevent these and other attacks, even by (stronger) MitM attackers, we recommend deployment of cryptographic defenses, in the common scenarios where the computational and communication overheads are acceptable.

REFERENCES

[1] S. M. Bellovin, "Security Problems in the TCP/IP Protocol Suite," Computer Communication Review, vol. 19, no. 2, pp. 32–48, apr 1989.
[2] CAIDA, "Anonymized Internet Traces 2012 Dataset," http:// www.caida.org/data/passive/passive 2012 dataset.xml, 2012.
[3] O. Gudmundsson and S. D. Crocker, "Observing DNSSEC Validation in the Wild," in SATIN, March 2011.
[4] H. Ballani, P. Francis, and X. Zhang, "A Study of Prefix Hijacking and Interception in the Internet," in ACM SIGCOMM Computer Communication Review, vol. 37, no. 4. ACM, 2007, pp. 265–276.
[5] R. Beverly, R. Koga, and K. Claffy, "Initial Longitudinal Analysis of IP Source Spoofing Capability on the Internet," Internet Society Article, 2013.
[6] S. M. Bellovin, "A Look Back at, "Security Problems in the TCP/IP Protocol Suite" in ACSAC. IEEE Computer Society, 2004, pp. 229–249.
[7] R. T. Morris, "A Weakness in the 4.2BSD Unix TCP/IP Software," AT&T Bell Laboratories, Tech. Rep., Feb. 1985. and Privacy, 2012, pp. 347–361.
[8] D. Kaminsky, "It's the End of the Cache as We Know It," in Black Hat conference, August 2008, http://www. blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/ BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf.
[9] M. Zalewski, "Strange Attractors and TCP/IP Sequence Number Analysis," http://lcamtuf.coredump.cx/newtcp/2001. [10] A. Herzberg and H. Shulman, "Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org," in CNS 2013. The Conference on Communications and Network Security IEEE, 2013.
[11] M. Zalewski, "A New TCP/IP Blind Data Injection Technique?" http://lcamtuf.coredump.cx/ipfrag.txt, 2003.
[12] Y. Gilad and A. Herzberg, "Fragmentation Considered Vulnerable," ACM Transactions on Information and System Security (TIS- SEC), vol. 15, no. 4, pp. 16:1–16:31, April 2013, a preliminary version appeared in WOOT 2011.
[13] A. Herzberg and H. Shulman, "Security of Patched DNS," in European Symposium on Research in Computer Security, ser. LNCS, S. Foresti, M. Yung, and F. Martinelli, Eds., vol. 7459. Springer, 2012, pp. 271–288. [Online]. Available: http://dx.doi.org/10. 1007/978-3-642-33167-1
[14] ——, "Vulnerable Delegation of DNS Resolution," in European Symposium on Research in Computer Security, ser. Lecture Notes in Computer Science. Springer, 2013.
[15] ——, "Socket Overloading for Fun and Cache Poisoning," in ACM Annual Computer Security Applications Conference (ACM ACSAC), December 2013.
[16] T. Shimomura and J. Markoff, Takedown: The Pursuit and Capture of Kevin Mitnick, America's Most Wanted Computer Outlaws - by the Man Who Did It, 1st ed. Hyperion Press, 1995.
[17] Klm, "Remote Blind TCP/IP Spoofing," Phrack magazine, 2007.
[18] ——, "When Tolerance Becomes Weakness: The Case of Injection- Friendly Browsers," in Proceedings of the International World Wide Web Conference, May 2013.
[19] Z. Qian, Z. M. Mao, and Y. Xie, "Collaborative TCP Sequence Number Inference Attack: How to Crack Sequence Number under a Second," in Proceedings of the ACM Conference on Computer and Communications Security. New York, NY, USA: ACM, 2012, pp. 593–604. [Online]. Available: http://doi.acm.org/10.1145/ 2382196.2382258
[20] Z. Qian and Z. M. Mao, "Off-Path TCP Sequence Number Inference Attack," in IEEE Symposium on Security and Privacy, 2012, pp. 347–361.