VoIP Malware: Attack Tool & Attack Scenarios

Mohamed Nassar, Radu State, Olivier Festor

VoIP Malware: Attack Tool & Attack Scenarios

Mohamed Nassar, Radu State, Olivier Festor
Centre de Recherche INRIA Nancy - Grand Est
615, rue du jardin botanique, 54602
Villers-Lès-Nancy, France
{nassar|state|festor}@loria.fr

***Abstract*— With the appearance of new Internet services like Voice over IP and IP television, malwares are in the way to update and extend their targets. In this paper, we discuss the emergence of a new generation of malwares attacking VoIP infrastructures and services. Such malwares constitute a real threat to the currently deployed VoIP architectures without strong security measures in place. We present one implemented environment that can be used to evaluate such attacks. Our "VoIP bots" support a wide set of attacks ranging from SPIT to DDoS and are tested against several VoIP platforms.**

I. Introduction

Malware is the general term grouping softwares designed to infiltrate or damage a computer system without the owner's informed consent. A worm is a special type of malware that can run on its own and propagate itself to other machines. An attacker using worms and other infection mechanisms to install bots on vulnerable machines is called bot herder or bot master. Upon their bootstrap, the compromised machines or Zombies connect themselves to their master to receive command input and execute attack operations. In many scenarios, the bot master rents time on the botnet (network of bots under a common control infrastructure) to a third party which orientates them to launch Distributed Denial of Service Attacks (DDoS) and massive SPAM. Botnets pose a severe threat to today's Internet. Over the recent years, botnets have become a popular and profitable market. Since April 2007 when the government and private web servers in Estonia were the target of a large DDoS, "cyberwar" is no more science fiction.

With the growing of new Internet technologies like VoIP, it is expected that botnets will play the same malicious role. VoIP services are well exposed to such threats since they are open and have to be reachable. Values of interest within a VoIP enterprise domain include signaling and media infrastructure, accounting directories, PBX services (voice mailboxes, gateways), individual user accounts, the internal networks running other services. Attacks can be transmitted across gateways to integrated networks like mobile and traditional telephony ones. Conversely, compromising VoIP applications constitutes a bridge to bypass security mechanisms and attack internal networks. Currently, some researchers argue that SKYPE - which is the most deployed peer to peer VoIP application- could be a backdoor [1]. SIP (Session Initiation Protocol - RFC 3261) -which is the de-facto standard signaling protocol- is a strong candidate to become the UFBP (Universal Firewall Bypass Protocol) or the universal payload injector. This is assessed after the discovery of many cross-site scripting (XSS)[1] and SQL injection attacks[2] due to SIP vulnerabilities. To be effective however, these vulnerabilities have to be exploited over a large distributed framework. A botnet seems perfectly adequate for this task.

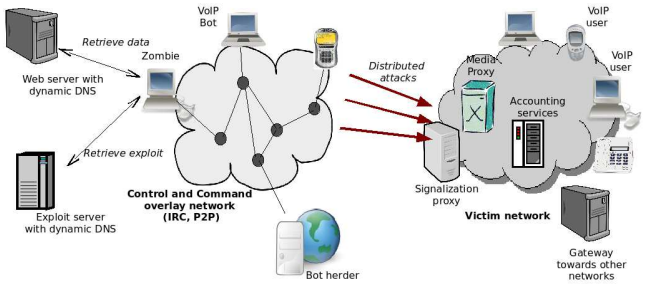

Fig. 1. VoIP Botnet Framework

In this paper, we present such a framework of VoIP bot and botnet in the context of SIP. Our bots are not equipped with propagation mechanisms so they don't represent a real threat value; rather than being used for research purposes. The overlay network of our framework is depicted in Figure 1.

The rest of the paper is organized as follow: In Section II we introduce the SIP protocol. We illustrate our bot architecture in Section III and we enumerate many attack scenarios in Section IV. We discuss implementation issues in Section V. In Section VI we debate the related work and we conclude the paper in Section VII.

II. SIP Background

SIP is emerging as the future standard for Internet telephony signaling because of the following strength points [2]:

- SIP is professionally developed by the IETF (Internet Engineering Task Force) to be scalable over Internet and utilizing Internet standards and capabilities (e.g. DNS, URL support, Wireless);
- SIP is text based with heritage from HTTP and SMTP, so it can be easily scripted, logged and inspected;

[1] http://seclists.org/fulldisclosure/2007/Oct/0174.html

[2] http://voipsa.org/pipermail/voipsec_voipsa.org/2007-October/002466.html

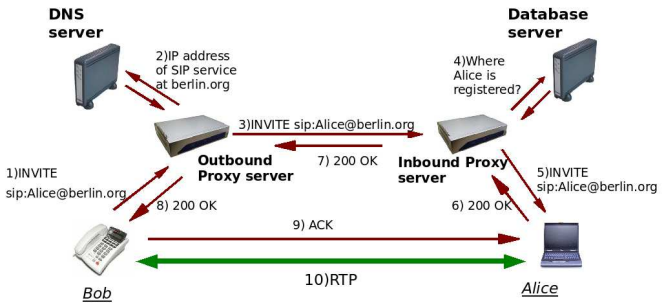

Fig. 2. SIP Call Establishment

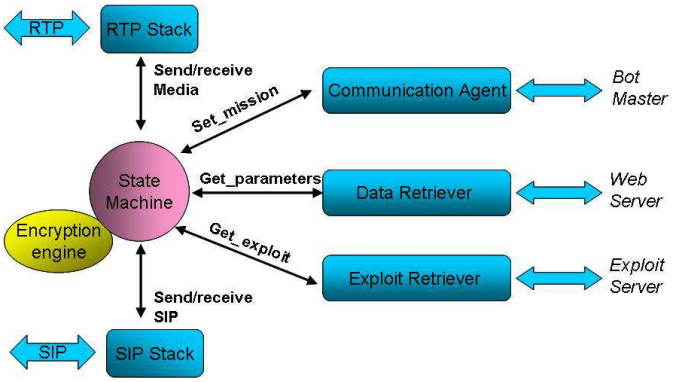

Fig. 3. VoIP Bot Architecture

- SIP has presence and instance messaging capabilities presenting a high potential to invent new applications in the future;
- SIP is well supported by the industry: it has been adopted by mobile operators in their third generation networks and services;
- SIP takes advantage of Internet security mechanisms as encryption, authentication, integrity and certificates, even if it inherits a large set of vulnerabilities from the underlying data networks from another side.

Basically, SIP allows two communicating parties to set up, modify and terminate a phone call. SIP is a request-response transaction-based protocol. A SIP Dialog is composed of one or more transactions. The SIP addressing scheme is based on URIs (Uniform Resource Identifier) e.g. `sip:user@host:port;parameters`. Proxy servers help routing SIP messages. A basic call initiation scenario is depicted in Figure 2.

III. VoIP Bot Architecture

Our bot architecture is shown in Figure 3. The stack of different protocols provides the bot with an application interface to use these protocols. The SIP stack is responsible for sending and receiving, manufacturing and parsing SIP messages. The RTP stack is responsible for coding and decoding, compressing and expanding, encapsulation and demultiplexing of media flows. Other stacks can be supported as well. For example, the STUN [3] protocol is useful to bypass NAT.

The communication agent allows the bot to exchange information and commands with the attacker. Most of the known botnets use IRC (Internet Relay Chat - RFC 1459) or peer-to-peer (P2P) networks for their control and command architecture. IRC is mainly designed for group communication and allows one-to-one communication (private discussion) as well. A channel (or a room) is supported by multiple servers building an application level spanning tree among them and relaying IRC messages between room visitors. P2P refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner [4]. Bot masters moved towards P2P networks because of their high degree of anonymity and privacy. For example in Freenet [5], when a peer sends a message, the peer identity is rewritten as the message is relayed among a chain of peers. Another example is the Lucent Personalized Web Assistant (LPWA) [6] which acts as a proxy server and allows consistent untraceable aliases for clients from servers. A LPWA client opens an account and is recognized upon returning to this account while his true identity is hidden from the server. The Slapper worm [7] which builds a P2P overlay network has advanced features like reliable end to end message delivery, coping with network partitions and reshaping, and anonymous message delivery.

The data retrieval component allows the bot to retrieve different kinds of data (e.g. list of VoIP extensions (URIs), advertising audio files, list of default passwords to try, SIP messages to shoot, etc ...) using a data communication protocol (e.g. FTP or HTTP Client). Web servers using a dynamic DNS server (i.e. the DNS changes the IP corresponding to the web server over time) are preferred to avoid being tracked.

The exploit retrieval component allows the bot to retrieve specific exploits against vulnerabilities and software flaws in VoIP products. The damage of such exploits ranges from remote DoS on the target (similar to a ping of death) to remote eavesdropping[3]. Cross script attacks and data base injection vectors can be carried by malformed SIP messages to attack embedded web servers in the targeted products and databases querying theses messages for accounting and statistics. Some exploits are stateless (consisting on shooting one SIP message) but others are stateful (based on the state machine of the target). Stateful attacks are formed by a series of messages where the content and the sending time of each message depends from the previous sent message and the corresponding reaction or response of the target to that message. For example, a stateful remote DoS on a Cisco 7940 SIP Phone[4] has been discovered using the KIF stateful fuzzer [8]. The bot master uses some format to describe stateful exploits and upload them to a server where they can be found by the bot. The bot parses the exploit description and builds a local state machine to perform the attack.

The encryption engine enables the bot to create digest

[3] `http://www.voipsa.org/pipermail/voipsec_voipsa.org/2007-August/002424.html`

[4] `http://www.voipsa.org/pipermail/voipsec_voipsa.org/2007-August/002422.html`

authentication from credentials when authentication is required in the process of an attack or an attempt of registration. Typical use of this engine is password cracking and CPU-based flooding against the target's authentication procedure.

The SIP state machine manages the operations of the bot with respect to the commands issued by the attacker. The `mission` of the bot as set by the attacker drives its behavior upon occurrence of SIP events (i.e. receiving a SIP request `RequestEvent` or a SIP response `ResponseEvent`) and `TimeOut` events. The transition from a state to another is constrained by predicates on a set of global and local variables. For example, when receiving a 200 OK message that belongs to some existing dialog, the bot's next step is based on the `Cseq_method` (which determines the method the 200 OK is in response for) and on the global attack parameters (mission, target IP, target SIP port ...).

Based on this architecture, different attack scenarios are possible as detailed in the next section.

IV. Attack Scenarios

SPIT

SPIT or SPAM over Internet Telephony refers to unsolicited calls intended for advertising or social engineering. Automated calls have already been used a couple of times like for example in the 2008 American presidential election.[5]

In a SPIT scenario, the attacker asks the bot to deliver an audio record to one or more destination URI. Similarly to e-mails being used in SPAM, URIs can be collected by web crawlers or in result to a VoIP domain enumeration. The bot manufactures an INVITE request carrying an SDP body, giving arguments like the destination URI, its IP address, its RTP port, the codec to be used and other media attributes. When the call is answered, the bot retrieves the audio record from the location as supplied by the attacker (e.g. from a URL or a local file in the compromised machine) and stream it to the callee.

Flooding

Flooding attacks target the signaling plane elements (e.g. proxy, gateway, etc.) with the objective to take them down or to limit their quality, reliability and availability. Flooding attacks can be categorized regarding their destination and their strategy. Whether the attack is destined to a valid URI in the target domain, a non existent URI in the target domain, a URI with an invalid domain or IP address, an invalid URI in another domain, or a valid URI in another domain, different damages are produced. The strategy is related to the nature of messages used during the attack: legitimate, malformed (carrying some exploits), invalid (non compliant to the SIP standard), and spoofed SIP messages (spoofed `From` and `Contact` header). Other attacks are targeted against the authentication process by using messages carrying valid `nonces` and requiring the target to compute digests which overwhelm its CPU capacity [9]. In a flooding scenario, the bot starts a thread which sends continuously request messages (INVITE or REGISTER) given the destination, the duration, the timing and the strategy as supplied by the attacker. Distributed flooding attacks can be easily organized by involving and synchronizing a number of bots.

Enumeration

Enumeration is the process of discovering valid SIP extensions (or URIs) in a SIP domain. Enumeration is usually preceded by a port scan to identify existent SIP proxies and user agents. The standard port used by SIP is 5060. The first step of enumeration is to identify if the SIP service is running on that port and what type and version of server is there. This is done by sending a simple OPTIONS message and interpreting its response. The searched information is usually found in the `Server` or the `User-Agent` header.

In an enumeration scenario, the bot retrieves a list of extensions/URIs from a specified location, then it probes them using INVITE, REGISTER or OPTIONS messages. OPTIONS enumeration is preferable since it is stealthier (it does not ring the phones, nor raise suspicions about the registration process). The bot has to match each request with the response it triggered based on the call-ID and/or transaction identifier. It analyzes the response to determine 1) if the dialed extension exists and is registered to the target, 2) exists and is temporarily unavailable, or 3) doesn't exist at all. The interpretation of responses depends on the target's type and version. For example, if the target carries an `OpenSER sip proxy (version 1.1.1-notls)` fingerprint, the response to an OPTIONS message destined to an extension is interpreted as follows: a "200 OK" means that the extension exists and it is registered, a "404 NOT FOUND" means that the extension is invalid, but a received "100 TRYING" before a final error response means that the extension is valid but not available for the moment.

Cracking

Remote brute force password cracking consists in repeatedly trying guesses for an account's password using INVITE or REGISTER requests. Unchanged default passwords of deployed VoIP platforms make them strongly vulnerable to such attacks. In a cracking scenario, the bot has to discover the registration or the voicemail password for a user name. Note that the user name used in the authentication process is not always the same as the one in the user's URI (which can be obtained by enumeration). The attacker has to know the user name or the voice mail extension before going to a brute force attack. The bot retrieves a list of default passwords corresponding to the target platform. For each guess, it sends a first REGISTER (or INVITE) asking for a challenge. An error response from the target gives him back a nonce. The bot calls its encryption engine to build a new request containing credentials based on the challenge and the temporal nonce. The target's final response decides if the guess was right or not.

[5] `http://edition.cnn.com/2008/POLITICS/10/23/robo.calls/index.html`

Fingerprinting

Remote fingerprinting allows the attacker to identify the type and the version of the SIP target platform. The simplest method consists on sending an OPTIONS message and extraction of the manufacturer string from its response. This process can be fooled if the manufacturer string was intentionally falsified. Smarter fingerprinting schemes as described in [10] or in [11] can be supported as well. In a fingerprinting scenario, the bot is asked to fingerprint one or range of SIP extensions. The bot has to send back its results to the attacker or -in order to not disclose him- to put them in a specified location where he can access them later.

Exploiting Specific Vulnerabilities

In an exploit attack scenario, the bot is either given the platform of the target or has to discover it by itself (by fingerprinting). The bot connects to an exploit server and retrieves a set of possible exploits corresponding to the target fingerprint. Each exploit should be tagged with some meta-data so the bot can choose the exploit which meets the attacker's aim. In case of stateful attacks, the bot builds a local attack state machine to execute the attack given local and remote parameters as reported by the attacker.

Interception and Modification

These attacks require an access to the internal network in the VoIP domain. If the attacker succeeds to compromise a machine inside the VoIP domain, many interception, eavesdropping, modification and man in the middle attacks are possible using ARP poisoning techniques. It is not just media and signaling traffic which is targeted, but also supporting protocols like DNS, DHCP, ICMP, and TFTP. In one scenario, if the bot knows the IP address of a phone and the IP address of the outbound SIP proxy, it can fool the phone into thinking that it is the proxy (by sending an RTP packet with the IP address of the proxy and the MAC address of the bot) and vice versa. Like that, the bot plays an MITM role by watching and forwarding messages between the two entities. Because typically no data integrity is deployed in current SIP implementations, the MITM can change an INVITE request before forwarding it. For example, it can redirect the call towards an IVR (Interaction Voice Response) to do Vishing (VoIP Phishing) scam.

Fraudulent Calls

The increasing financial and informational value of VoIP will attract more Internet hackers into attacking VoIP middlewares, take control over them and execute remote code. Future scenarios are to ask the compromised phone to dial an overtaxed number (similarly to the modem-based dialers in the near past) or to record all calls. Terms like "VoIP dialer" or "VoIP logger" are going to appear in the near future. A one million dollar idea arises immediately: let a large number of victims (one million) dial an overtaxed number resulting on only one additional dollar on the monthly bill of each caller. Most of the victims will not complain about.

Fig. 4. Screen view of the bot master

Propagation

Propagation models of worms exploiting VoIP and mobile vulnerabilities and using directories in the compromised platforms to spread up are discussed in [12]. By its fingerprinting, exploit retrieving and executing capabilities, our bot constitutes a basis for such a worm using an algorithm like the following one:

```
Forall uri in PhoneBook
 fingerprint = Bot.Fingerprint(uri);
 exploit = Bot.
           retrieve_exploit(fingerprint);
 Bot.send_exploit (exploit, uri);
 Bot.upload_version(uri);
End_Forall
```

V. Implementation and Experiments

We implemented a proof-of-concept worm-free IRC bot based on SIP and RTP using the JAVA language. For SIP we used the Jain SIP library [13]; for RTP we used the JMF library[14] and for IRC we used the PircBot library[6]. Our code is available under an open source license [15]. The bot is currently able to perform DoS, SPIT, SCAN, CRAK, FINGEPRINT, SHOOT, EXPLOIT and REGISTER functionalities. Moreover, the bot master is able to perform a collective suicide of all the bots. The screenshot of Figure 4 shows the IRC client (Xchat[7]) of the bot master upon the connection of one bot.

Deployed on an Intel Pentium 4 CPU 3.40GHz and 2G RAM memory machine running a Linux kernel 2.6.18-1, the

[6] http://www.jibble.org/javadocs/pircbot/index.html
[7] http://www.xchat.org/

bot is able to send around 10,000 messages per second with different call-Ids. The call-ID seed is the number of the bot as set by the attacker. Messages from different bots have different Call-Ids. We used a similar machine with 3G RAM memory to be the target (hosting Asterisk and OpenSER[8]). Using legitimate messages and non existent URI destination, one bot is able to raise the target CPU to 100% in case of both Asterisk and OpenSER, and 2 bots are able to saturate the bandwidth of a LAN connection (about 12 MBytes/s). Asterisk consumes 25% of the host system memory (i.e. 750 MB) after 100 seconds of attack (i.e. 0.25% raise in memory/s), while OpenSER memory consumption depends on the number of child processes as configured by the administrator (Each child reserves a 33 MB memory space). OpenSER integrates a defense module against flooding attacks called PIKE. PIKE blocks an IP address for a period p after receiving a number Th coming from that address during a prefixed sampling period s.

In order to perform a flooding against a PIKE-protected OpenSER, a number of bots should be synchronized. The first step is to discover the PIKE parameters. We propose the scheme of Figures 5 and 6.

We need only $1+p/s$ bots to continuously deny the service at the proxy. A first bot starts the attack and is blocked after the first sampling period, the other p/s bots, each by turn, assure the attack until the first bot is unblocked again. For sake of simplicity, a scheme using only three bots is depicted in Figure 7.

Several flooding mitigation mechanisms are currently proposed ([16], [17], [18], [19]). However, our bots are able to bypass such mechanisms at least from a theoretical point of view. Threshold-based systems can be deceived by fine-tuned the flooding rates. Learning-based systems can be "poisoned" by inserting specially crafted data points so that the model of "normality" drifts into the direction of an attack vector [20].

[8]The project has evolved in two parallel projects OpenSIPS and Kamailio

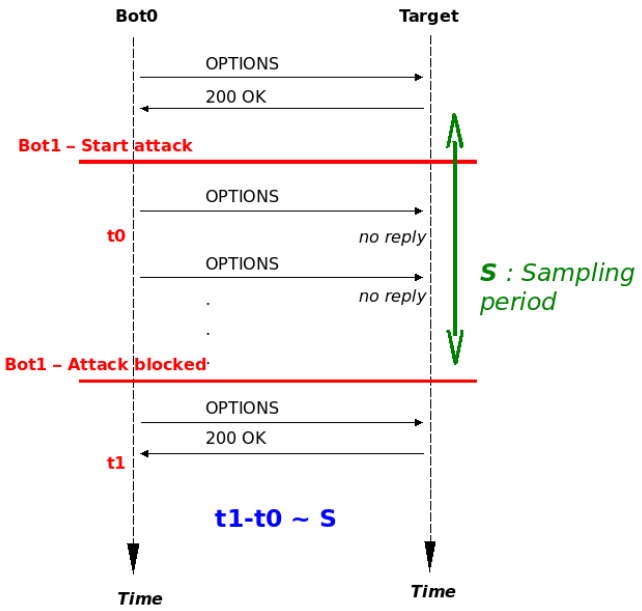

Fig. 5. Discovering the s Parameter

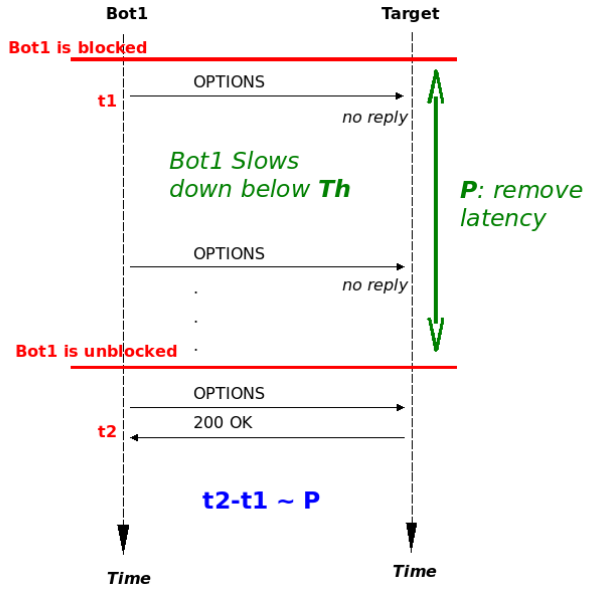

Fig. 6. Discovering the p Parameter

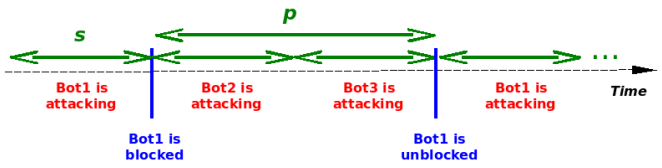

Fig. 7. Attacking a PIKE-protected system

VI. Related Works

Computer viruses and malwares [21] pose a serious threat to the computer network infrastructures of our society. Logic bombs, Trojan horses, backdoors, spywares and worms should be studied for their different aspects to enhance our protection, intrusion detection and prevention. With the emergence of VoIP, many vulnerabilities have been discovered through security testing of SIP implementations [22] that can be exploited by such malwares.

Appropriate security approaches have been well investigated to increase the immunity of this new technology. An early warning system has been published in [23]. A holistic intrusion detection solution based on event and alert correlation is described in [24]. This paper uses a different approach since it adopts the attacker's perspective and therefore provides an assessment tool to feedback currently deployed security architectures.

A number of similar tools are already available[9]. SIPp[10] is a

[9]http://www.voipsa.org/Resources/tools.php
[10]http://sipp.sourceforge.net/

SIP traffic generator providing basic call flow scenarios while allowing more complex scenarios to be described using XML. Sipsak[11] is a command line tool that can be used for simple SIP tests. C07-SIP[12] is a byproduct of the PROTOS project providing a group of tests against SIP parsers. SIPSendFun[13] is a set of php scripts to send spoofed SIP messages. Skora[14] wrote several Perl scripts demonstrating different attacks such as BYE and CANCEL attacks. Our VoIP bot is novel with respect to these tools because of its numerous functionalities and the ability it gives to manage distributed attacks.

VII. Conclusion

In this paper, we presented a framework of VoIP-specific malware using bots installed on compromised machines and we showed how different attack scenarios can be supported by such a framework. We presented a prototype implementation that we experimented against VoIP servers and showed how cooperating bots can be synchronized to bypass flooding defense mechanisms. Future works include the investigation of efficient and scalable defense mechanisms against VoIP malwares and distributed denial of service attacks.

References

[1] C.-M. Leung and Y.-Y. Chan, "Network forensic on encrypted peer-to-peer voip traffics and the detection, blocking, and prioritization of Skype traffics," in *WETICE '07: Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 401–408.

[2] A. Johnston, *SIP: Understanding the Session Initiation Protocol, Second Edition*. Norwood, MA, USA: Artech House, Inc., 2003.

[3] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN – Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs)," Internet Engineering Task Force: RFC 3489, March 2003.

[4] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja1, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer computing," July 2003, hPL-2002-57 (R1).

[5] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: a distributed anonymous information storage and retrieval system," in *International workshop on Designing privacy enhancing technologies*. New York, NY, USA: Springer-Verlag New York, Inc., 2001, pp. 46–66.

[6] E. Gabber, P. B. Gibbons, D. M. Kristol, Y. Matias, and A. Mayer, "Consistent, yet anonymous, web access with LPWA," *Commun. ACM*, vol. 42, no. 2, pp. 42–47, 1999.

[7] I. Arce and E. Levy, "An analysis of the slapper worm," *IEEE Security and Privacy*, vol. 1, no. 1, pp. 82–87, 2003.

[8] H. J. Abdelnur, R. State, and O. Festor, "KiF: a stateful SIP fuzzer," in *IPTComm '07: Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications*. New York, NY, USA: ACM, 2007, pp. 47–56.

[9] M. Luo, T. Peng, and C. Leckie, "CPU-based DoS attacks against SIP servers," in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2008)*. IEEE, April 2008.

[10] H. Yan, K. Sripanidkulchai, H. Zhang, Z.-Y. Shae, and D. Saha, "Incorporating active fingerprinting into SPIT prevention systems," in *Third annual security workshop (VSW'06)*. ACM Press, Jun 2006.

[11] H. Abdelnur, R. State, and O. Festor, "Advanced network fingerprinting," in *RAID '08: Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*. London, UK: Springer-Verlag, 2008, pp. 311–330.

[11] http://sipsak.org/

[12] http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/

[13] http://www.security-scans.de/index.php?where=ssf

[14] http://skora.net/voip/voip.html

[12] C. Fleizach, M. Liljenstam, P. Johansson, G. M. Voelker, and A. Mehes, "Can you infect me now?: malware propagation in mobile phone networks," in *WORM '07: Proceedings of the 2007 ACM workshop on Recurring malcode*. New York, NY, USA: ACM, 2007, pp. 61–68.

[13] P. O'Doherty and M. Ranganathan, "JAIN SIP Tutorial: Serving the developer community," http://www-x.antd.nist.gov/proj/iptel/tutorial/JAIN-SIP-Tutorialv2.pdf.

[14] "Java Media Framework API Guide," SUN Microsystem, november 1999, http://java.sun.com/products/java-media/jmf/2.1.1/guide/index.html.

[15] "The VoIP Bot project," http://gforge.inria.fr/projects/voipbot/.

[16] B. Reynolds and D. Ghosal, "Secure IP Telephony using Multi-layered Protection," in *Proceedings of The 10th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, feb 2003.

[17] E. Chen, "Detecting DoS attacks on SIP systems," in *Proceedings of 1st IEEE Workshop on VoIP Management and Security*, San Diego, CA, USA, apr 2006, pp. 53–58.

[18] S. Ehlert, C. Wang, T. Magedanz, and D. Sisalem, "Specification-based denial-of-service detection for sip voice-over-ip networks," *icimp*, vol. 0, pp. 59–66, 2008.

[19] H. Sengar, H. Wang, D. Wijesekera, and S. Jajodia, "Detecting VoIP Floods using the Hellinger Distance," *Transactions on Parallel and Distributed Systems : Accepted for future publication*, sep 2007.

[20] B. Nelson and A. D. Joseph, "Bounding an attack's complexity for a simple learning model," in *In the Proceedings of the First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, June 2006.

[21] J. Aycock, *Computer Viruses and Malware (Advances in Information Security)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[22] C. Wieser, M. Laakso, and H. Schulzrinne, "Security testing of sip implementations," Tech. Rep., 2003.

[23] M. Nassar, R. State, and O. Festor, "Voip honeypot architecture." in *Integrated Network Management*. IEEE, 2007, pp. 109–118.

[24] M. Nassar, S. Niccolini, R. State, and T. Ewald, "Holistic voip intrusion detection and prevention system," in *IPTComm '07: Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications*. New York, NY, USA: ACM, 2007, pp. 1–9.