

DRAPA - a flexible framework for evaluating the quality of VoIP components

Bradley Clayton, Alfredo Terzoli and Barry Irwin

Department of Computer Science

Rhodes University, Grahamstown, South Africa

Tel: 046 6038291 Fax: 046 6361915

g01c2974@campus.ru.ac.za, a.terzoli@ru.ac.za, b.irwin@ru.ac.za

Abstract—When adding to or altering a VoIP system, the overall performance and quality of the system is at risk. For example, adding confidentiality, integrity and authentication (CIA) would incur an overhead for each additional security method. A method of measuring the performance of a VoIP system after a change or addition is needed. This paper describes a framework and testbed (DRAPA) which provides a flexible base from which VoIP performance analysis systems can be built. DRAPA generates and collects data from any part of a VoIP system within a real domain. This paper also discusses the flexibility of DRAPA. While security is our primary focus, DRAPA allows the user to configure the testbed and change the type and nature of data to be collected.

Index Terms—VoIP, VoIP Security, VoIP Performance analysis

I. INTRODUCTION

OUR primary research focuses on the addition of confidentiality, integrity and authentication (CIA) to real-time multimedia [1]. Our specific focus is to add security to Asterisk, an open source software PBX, based systems [2]. This research called for a method of measuring the performance and quality after a security method was implemented into Asterisk. Data must be collected before any quality analysis can be done and a testbed, which we named DRAPA (the Distributed Real-time Application Performance Analyser), was designed and implemented to accomplish this. DRAPA is a framework and flexible implementation which allows us to generate and collect data from any part of a VoIP system within a real domain. Collecting data within a real domain distinguishes DRAPA from other research efforts [3] which use simulations to generate data.

Although the focus of our study is on the performance aspect of security and VoIP, DRAPA is designed to be used to analyse any area of VoIP. For example, the performance of codecs or trunking protocols could be investigated.

Depending on the nature of the data collected from DRAPA, several methods of analysis already exist. For example, Mean Opinion Score (MOS) analysis [4] is a subjective scoring technique whereby human participants are asked to listen to recordings generated by the testbed. The participants rate the

This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom SA, Business Connexion, Comverse, Verso Technologies, Tellabs, StorTech, THRIP and the National Research Foundation and the German Academic Exchange Service (DAAD).

quality of each recording on a scale of 1 to 5 where typically 1 is unsatisfactory and 5 is excellent. Automated methods of MOS analysis exist [3], where algorithms are able to produce a MOS score, allowing one to perform MOS analysis without needing human resources.

The next section introduces the overall architecture of DRAPA. Section III describes agents distributed within DRAPA. The distributed agents aid in the core control of DRAPA which is explained in section IV. The flexible component of DRAPA, action modules, are described in section V. A walk-through example of data collection and analysis follows in section VI. Lastly, a web front-end, used to display the current state of DRAPA, is described in section VII.

II. DRAPA'S ARCHITECTURE

DRAPA consists of five entities: endpoints, a VoIP server, a traffic shaper, a web interface and a centralized controller. This section will discuss the architecture and how the entities are related to one another.

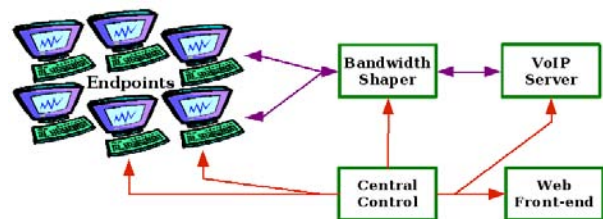


Fig. 1. Distributed Real-time Application Analyser (DRAPA) physical architecture

As mentioned in section I, DRAPA performs data capture in a real domain rather than a simulated one. Therefore, physical resources are required so that a complete system can be implemented before data collection begins. The first resource needed are endpoints. Due to a high availability of computer laboratories at Rhodes University, DRAPA uses these laboratory machines as endpoints. The laboratory machines are therefore a shared resource between DRAPA and students. Collected data cannot be relied upon if a machine is being used by DRAPA and a person in the laboratory at the same time. This creates an important design constraint: DRAPA can only make use of idle machines. The exclusion of used endpoints

is explained in section III. To make a normal machine act as an endpoint, each machine runs Asterisk on top of the Linux operating system. (DRAPA is capable of integrating with other endpoint software, for example a softphone.)

The second component is a VoIP server which also runs Asterisk on top of the Linux operating system. Asterisk supports a large range of VoIP protocols which allows us to broaden our research within a set environment. During a test, real-time streams are created between the endpoints but are routed via the VoIP server.

The third component of DRAPA, the traffic shaper, is created by placing a traffic shaping node between the VoIP Server and networked endpoints. Our traffic shaping node is a Soekris Single Board Computer (SBC) running the FreeBSD kernel re-compiled to enable the Internet Protocol Firewall (IPFW), bridging and Dumynet [5]. This combination allows us to realise a variety of network conditions. For example, we are able to drop packets, create latency, create duplicate packets and limit bandwidth. The traffic shaping node is also utilized during the data collection process, for example recording the rate of data sent to and from the VoIP server. Data collection is explained further in section VI.

The web interface (which is explained in more detail in section VII) provides a window into the current state of DRAPA.

The last entity of DRAPA is the centralized controller. The controller can be broken into three components. The core of the controller is tasked with coordinating the actions of the endpoints, the VoIP server and the traffic shaper. We refer to these responsibilities as the *central management system*, which is explained in section IV. The second component of the controller provides the flexibility of DRAPA's customization. Customization is achieved through the implementation of user configurable pluggable *action modules*. A pluggable action module contains instruction sets, in software, which are used to control each entity of the testbed. The pluggable action modules are explained in section V.

To aid in control, the controller makes use of distributed agents which report on and manage the endpoints. This third component is discussed in the next section.

III. DRAPA'S DISTRIBUTED MANAGEMENT SYSTEM

To maintain control of the endpoints, each is tagged with a state. Endpoints are always in one of the following three states:

- AVAILABLE - The endpoint is online and ready to be used in a test;
- WORKING - The endpoint is online and currently used in a test;
- USED - The endpoint is online but is being used by a person in the laboratory.

Note that an endpoint which is offline (because it is not switched on, or is booted into Windows instead of Linux) is not registered in the list. Therefore, an OFFLINE state is not required.

Two components, the DRAPA daemon (DRAPAD) and DRAPA slave (DRAPAS), form the distributed management

system. The daemon and slave focus solely on the endpoints and share two tasks. Firstly, they are responsible for maintaining a register of endpoints and their states. Secondly, they ensure that each AVAILABLE endpoint is running up-to-date DRAPA software.

The daemon can be run as an independent entity within the DRAPA testbed. However, it is acceptable to run the daemon in conjunction with the controller on a single machine, which is referred to as the Test Management Server (TMS). The daemon iteratively executes the following actions:

- 1) Upon start up, the daemon queries all endpoints in the AVAILABLE state to ensure that they are still online. An endpoint which does not respond is removed from the list of registered endpoints. (Well behaved endpoints usually de-register with the TMS when shutdown.)
- 2) While checking AVAILABLE endpoints, the daemon ensures that they are up-to-date with the latest version of the slave agent.
- 3) Lastly, the daemon searches predefined IP ranges for hosts which may have not registered properly when coming online. Any hosts found are registered as an online endpoint and their state is set to AVAILABLE unless they are in use. After this search, the daemon begins its cycle again with the first action.

The slave component is an agent which is run on each endpoint. The slave performs the following tasks:

- 1) informs the TMS when a endpoint comes online or goes offline,
- 2) informs the TMS when a endpoint is in use by a person in the laboratory,
- 3) ensures that all testing processes are closed down should a person begin to use the endpoint.

When the TMS is alerted to any of the above events, an appropriate action is taken. For example, if a laboratory host is currently incorporated in a test and somebody logs into the machine, DRAPA discards the current test. This functionality enables DRAPA tests to coincide with student laboratory usage. Should a test be discarded, DRAPA restarts the test ensuring that any laboratory machine in use is not incorporated into the new test.

The centralized controller uses the list of registered endpoints. The controller selects endpoints registered with an AVAILABLE state to use during a test cycle. The controller is described in the next section.

IV. DRAPA'S CENTRAL MANAGEMENT SYSTEM

The logic behind DRAPA is contained within the DRAPA controller (DRAPAC). The controller is responsible for maintaining control of the endpoints, VoIP server and traffic shaper. The controller also makes decisions in order to effectively utilize available resources and maintain an even distribution of data. The following lists the actions made by the controller:

- 1) Firstly, the TYPE of test to be performed is set. The TYPE is used to differentiate among data collected. For example, one set of data may be identified by a TYPE set to *SRTP* while another is set to *GSM*. In this example, the former TYPE relates to data collected while testing

the Secure Realtime Transport Protocol [6] while the latter refers to the GSM [7] codec.

- 2) The controller continues by taking into account the number of endpoints online which are in the AVAILABLE state. We shall refer to the number of available endpoints as x . If x is even and greater than any previously recorded number for the set TYPE, all endpoints are used to create $x/2$ calls, via the VoIP server. If x is lower than the largest recorded number of calls for the set TYPE, the controller searches the collected data to find a record generated from y number of calls, where y suits two conditions. Firstly, $0 < y < (x/2) + 1$ and secondly, the data associated with y number of calls appears the least number of times. In other words, we are looking for a number of calls which has been tested the least. If y for all data records for the current TYPE are within a range of two, the controller randomly selects a number of calls to create.
- 3) After deciding on y , the controller begins to execute sections of the action plug-in, explained in detail in section V. The action plug-in contains instructions which are used to create calls.
- 4) The controller waits 60 seconds for data to be generated. Then the controller ensures that y calls still exist. If there has been a problem and calls were lost, the current test is discarded.
- 5) If the number of existing calls is correct, the controller ends the calls and begins to collect data from nodes within the testbed. Data collection and the database are explained in section VI.

DRAPA is flexible because the methods used to manage calls, collect data, interact with a database and perform checks are customizable. The next section describes the pluggable action modules which perform these actions and are configurable by the user.

V. DRAPA'S PLUGGABLE ACTION MODULES

DRAPA's flexibility lies in its pluggable action modules. A single module defines all the instructions needed to control nodes within a DRAPA testbed. A module is made up of functions. Each function defines a set of instructions which, when called by the controller, perform a specific task within the testbed.

Action modules are written in Perl [8] and are designed to be easily configurable by the user. The following are examples of functions within a pluggable action module:

- CREATE_CALLS defines a set of actions to be taken when setting up a call between two endpoints. This function is called once for every new call and passed the host names for the caller and callee endpoints,
- STOP_CALLS describes the actions needed to stop calls previously created by CREATE_CALLS,
- TEST_CALLS should return *true* or *false*, depending on whether the calls created still exist or not. This function is used to check that calls have not been dropped during a test cycle.

The ability to reconfigure functions within a module allows us to perform different tests within a single problem domain.

For example, one testing session can focus on the performance when using the Secure Real-time Transport Protocol [6] while the next can focus on the performance when using IPSec [9]. This allows us to produce multiple data sets, from a realistic but controlled environment, all of which can be compared to each other. The entire problem domain can be re-configured by replacing the pluggable action module. For example, we could re-configure DRAPA to make use of a softphone on the endpoints instead of Asterisk.

VI. COLLECTION AND INITIAL PROCESSING OF STATISTICAL DATA

As discussed in section V, the data collected and the way in which it is processed is up to the user. The INIT_COUNTERS, FINAL_COUNTERS and DO_MATH functions, within a pluggable module, describe how data is collected and processed. In this section we discuss only two examples of data collection and processing, namely CPU and total bandwidth usage. It should be noted that in normal circumstances other data collected include jitter buffer size, latency, bandwidth in versus out and number of key re-exchanges (when working with security protocols).

To begin, the method for collecting data is defined in the INIT_COUNTERS and FINAL_COUNTERS functions. In this example, we can make use of SNMP or simply SSH remote command requests to the VoIP server for CPU information and the traffic shaper for bandwidth information. The difference between the INIT_COUNTERS and FINAL_COUNTERS functions is merely about where results should be stored in the database, so that a *before* and *after* comparison can be made.

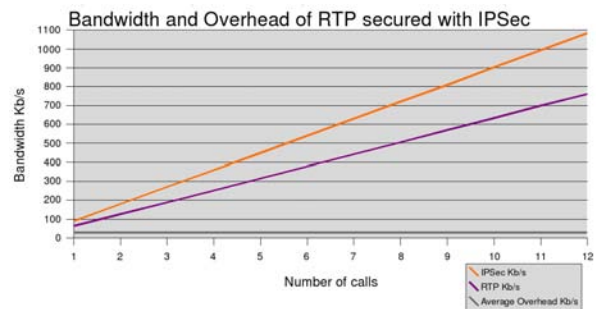


Fig. 2. Bandwidth used, with and without IPSec securing RTP traffic

After a test cycle has been completed, the actions in DO_MATH are performed. In our example, we subtract the initial from the final data readings and write the result to the database. Figure 2 graphs the result of a similar test, where the bandwidth used by a VoIP system secured with IPSec was assessed using DRAPA. To achieve this, two pluggable action modules were created and named IPSEC and RTP DRAPA tags data added in the database with the module name in order to distinguish between tests. Data tagged with IPSEC refers to data collected with VoIP calls were secured with IPSec. RTP tagged data corresponds to data collected when calls were not secured at all. This data, between 1 and 12 calls, was extracted from the database and used to draw the graph in Figure 2. From the graph we are able to relate the bandwidth usage

to theory [10] and realise that IPSec adds 26Kb/s, a 42% overhead, to the RTP traffic. This is a significant overhead unless security is of utmost importance.

VII. DRAPA'S GRAPHIC FRONT END.

DRAPA features a graphical web interface, shown in figures 3 and 4. The interface displays three areas of information. Figure 3 displays the currently loaded pluggable action module. The Stat distribution table displays the distribution of collected data over the number of calls.

Current type of test is Testing

Stat distribution:	
Call Count	Stats Collected
12	1
6	296
10	296
8	296
9	296
5	297
7	297
11	297
2	1162
3	1162
4	1162
1	4548

Fig. 3. DRAPA's web-interface - endpoint status table

The DRAPA controller attempts to keep the distribution of data as close together as possible but will still collect data when endpoints are available. For this reason figure 3 above shows that 4548 tests have been done for one call while only one test has been done for twelve calls. This is because only two endpoints are needed for a test of one call but twenty four are needed for a test of twelve calls.

Online endpoint status:			
Host	State	Call Type	Remote Peer
ug101.ict.ru.ac.za	Used		
ug107.ict.ru.ac.za	Available		
ug125.ict.ru.ac.za	Working	caller	ug133.ict.ru.ac.za
ug132.ict.ru.ac.za	Available		
ug133.ict.ru.ac.za	Working	callee	ug125.ict.ru.ac.za
ug141.ict.ru.ac.za	Available		
ug142.ict.ru.ac.za	Used		
ug147.ict.ru.ac.za	Available		
ug149.ict.ru.ac.za	Available		

Fig. 4. DRAPA's web-interface - test type and data distribution

The Online endpoint status table (figure 4) lists the address and state of each online endpoint. Offline endpoints are not listed in the table. When an endpoint is in the WORKING state, the call type and remote peer fields are also displayed. The call type represents whether the host is the caller or callee. The remote peer notes which endpoint is on the other side of the call.

VIII. CONCLUSION

DRAPA allows us to collect data from VoIP systems within a real domain. Currently, DRAPA is being used to generate data from an Asterisk-based VoIP system with added security. The data is analysed to assess the performance, and deterioration in quality, as a result of security overheads. However,

through the use of pluggable action modules and functions, DRAPA can be configured to generate and collect data from other VoIP problem domains.

A possible extension is to incorporate the flexibility and user control of DRAPA into the web-interface. This would allow a user to perform current command line tasks through a graphic interface. For example, pluggable modules could be loaded and the testbed could be paused and resumed via the web-interface.

REFERENCES

- [1] B. Clayton, A. Terzoli, and B. Irwin, "Performance Cost in Securing Confidentiality, Integrity and Authenticity of VoIP Communications," in *Proceedings of SATNAC 2005 - Convergence - Can technology Deliver*, September 2005. [Online]. Available: <http://www.cs.ru.ac.za>
- [2] J. Penton and A. Terzoli, "Asterisk: A Converged TCM and Packet-based Communications System," in *Proceedings of SATNAC 2003 - Next Generation Networks*, September 2003. [Online]. Available: <http://www.cs.ru.ac.za>
- [3] T. A. Hall, "Objective Speech Quality Measures for Internet Telephony," in *Proceedings of SPIE*, vol. 4522, 2001. [Online]. Available: <http://www.amtd.nist.gov/>
- [4] J. G. Beerends, A. P. Hekstra, A. W. Rix, and M. P. Hollier, "Perceptual Evaluation of Speech Quality (PESQ), the new ITU standard for end-to-end speech quality assessment. Part II - Psychoacoustic model," ITU-T, Tech. Rep., October 1998. [Online]. Available: www.tollyresearch.com
- [5] W. A. Vanhoner, "Evaluation of the FreeBSD dummynet network performance simulation tool on a Pentium-4 based Ethernet Bridge," Swinburne University of Technology - Center for Advanced Architectures, Tech. Rep., December 2003. [Online]. Available: www.epfl.ch
- [6] D. McGrew, E. Carrara, M. Baugher, M. Naslund, and K. Norrman, "RFC 3711: The Secure Real-time Transport Protocol (SRTP)," Cisco Systems, Inc and Ericsson Research, Tech. Rep., March 2004. [Online]. Available: <http://www.ietf.org>
- [7] J. Scourias, "Overview of the Global System for Mobile Communications," University of Waterloo - Shoshin Research Group, Tech. Rep., October 1997. [Online]. Available: <http://ccnga.uwaterloo.ca/~jscouria/GSM/gsmreport.html>
- [8] J. Bowls and S. Vance, "Scripting with Perforce," Swinburne University of Technology - Center for Advanced Architectures, Tech. Rep., April 2005. [Online]. Available: <http://www.perforce.com/>
- [9] C. R. Davis, *IPSec. Securing VPNs*. Berkeley, California: Osborne/McGraw-Hill, 2001.
- [10] N. Networks, "White paper - voip bandwidth calculation," Castlegate Business Park - Monmouthshire - United Kingdom, Tech. Rep. [Online]. Available: [urlhttp://www.newport-networks.com](http://www.newport-networks.com)

Bradley Clayton is a student at Rhodes University in the Department of Computer Science. He is currently reading for his Masters in Science and has a keen interest in VoIP and Computer Security

Alfredo Terzoli is the Project Director for the Centre of Excellence (CoE) in Distributed Multimedia at Rhodes University and the Project Director for the CoE in E-Commerce and E-Learning at the University of Fort Hare. His research interests include real-time multimedia over packet networks and ICTs in rural development.

Barry Irwin currently works in the Computer Science Department at Rhodes University. His research interests include Firewall Technologies, Network Traffic management, Applications of Cryptograph