RTC-TIE

Threat intelligence Exchange

Alexandr Dubovikov

Senior Voice Architect and Designer at **QSC AG**

Lead Developer	**HOMER**
Founder	**SIPCAPTURE**
Co-Founder, CTO	**QXIP BV**

Warning!

This is a true request for criticism from our community - not a complete solution!

The ideas and proposals expressed in this presentation are fishing for comments and input from all those experienced and passionate about solving the same challenges on a daily basis, to incorporate the best concepts and make them available for all

Our ultimate goal is to create a distributed layer where we can all protect our systems from common enemies, leaving each individual, company or group freedom in terms of managing local vs. shared attack resources being pushed to the community.

Fraud Attacks.

The problem needs no introduction - "**Bad Guys**" are out there diligently and perpetually probing all of our exposed VoIP Infrastructure ***24/7/365*** looking for gaps, accounts, routes they can exploit for **criminal profit** at the expense of your business.

Most operators can defend themselves - sometimes spectacularly *(like QSC or SipGate)* other times… not really!

On a daily basis millions of packets and cycles are wasted by *scanners* and as millions of real Euros are *burned* by Fraud

Smaller users are the most affected, left no "*default*" way to protect their setups full of defaults and exploitable configs.
John Doe deserves a default blacklist option for his little home PBX, and so does Johnny Operator 5 for all its honest customers!

Hello? Solutions Already Exist.

Meh. Not Really.

Classic Blacklists are boring and old fashioned, often working in a localized, non coordinated fashion.

Parsing cdrs and logs feeding fail2ban and local only iptables rules is not flexible and hardly distributed, if ever

Local Only detections are selfish and wasteful method only temporarily addressing the real problem.

Imagine if any confirmed Attack source ban on a federated network could automatically propagate to your systems

Post Mortem detections via CDR are futile, ongoing breaches cannot be detected or halted fast enough.

Everything is real-time, including Fraud. Attack Patterns, Botnets and Scanners can and should be dealt with as such.

Existing Blacklists are mostly controlled by private companies without transparency, not Communities.

Alternatives.

Running a Local, Private Blacklist

Using tools such as *fail2ban* and others, block your own attackers across your network *(IF you can detect them)*

Pro:	Cheap to run
Cons:	Hard to distributed
	Depend on plain logs or CDRs

Using an External, Public Blacklist

There are a few public options available options out there such as VoIPBL and others available to use

Pro:	Lazy Factor 100%
Cons:	Controlled by private entities. Who's who?
	Lack for native community Integrations

Open Gap. Open Source.

The **OSS VoIP** ecosystem is notorious for providing innovative solutions working in harmony with each other, very often surpassing solutions provided by gigantic vendors with the direct drive of great Humans, behind every good innovation.

This is sadly *not* yet the case for Voice *Fraud Prevention*, where defense is sometimes considered accessory - until it's too late.

This is where idea for an **Distributed Blacklist** for exchanging **Real-Time Communication** focused **Threat Intelligence** *is born*

The propositive key features of our project should be:

- Real-Time, just like our communications
- Distributed, without complex configurations
- Offline-First, always ready for its primary goal
- Open-Source & Community Maintained, Trusted
- Backed and Fueled by reputable Operators sharing detections

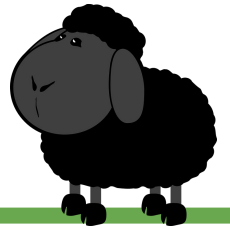
Getting our Hands Dirty: CaCheep.

In order to quickly prototype the concept, we've slashed together **CACHEEP** - a smart **LRU-Cache** providing the basic functionality and first building block we needed to create and demonstrate our ***offline-first, distributed blacklist*** concept.

Cacheep is only designed to be the *"local layer"* where queries can be answered with *low resource and low latency* ideal for RTC

A simple REST API to program self-expiring entries, optionally stored/distributed via pluggable backends *(Redis, GunDB)* and consumable by clients via DNS/ENUM or plain REST to drive routing decisions, correlation pairings and much more.

Behind the scenes, data is seamlessly replicated over to clients via aggregation servers or p2p via websocket connections, guaranteeing a consistent offline-first consumption of data and fast propagation of events for nodes to use and enforce.



Distributed.

Cacheep can currently use **Redis** as backend to store and further distribute its data.

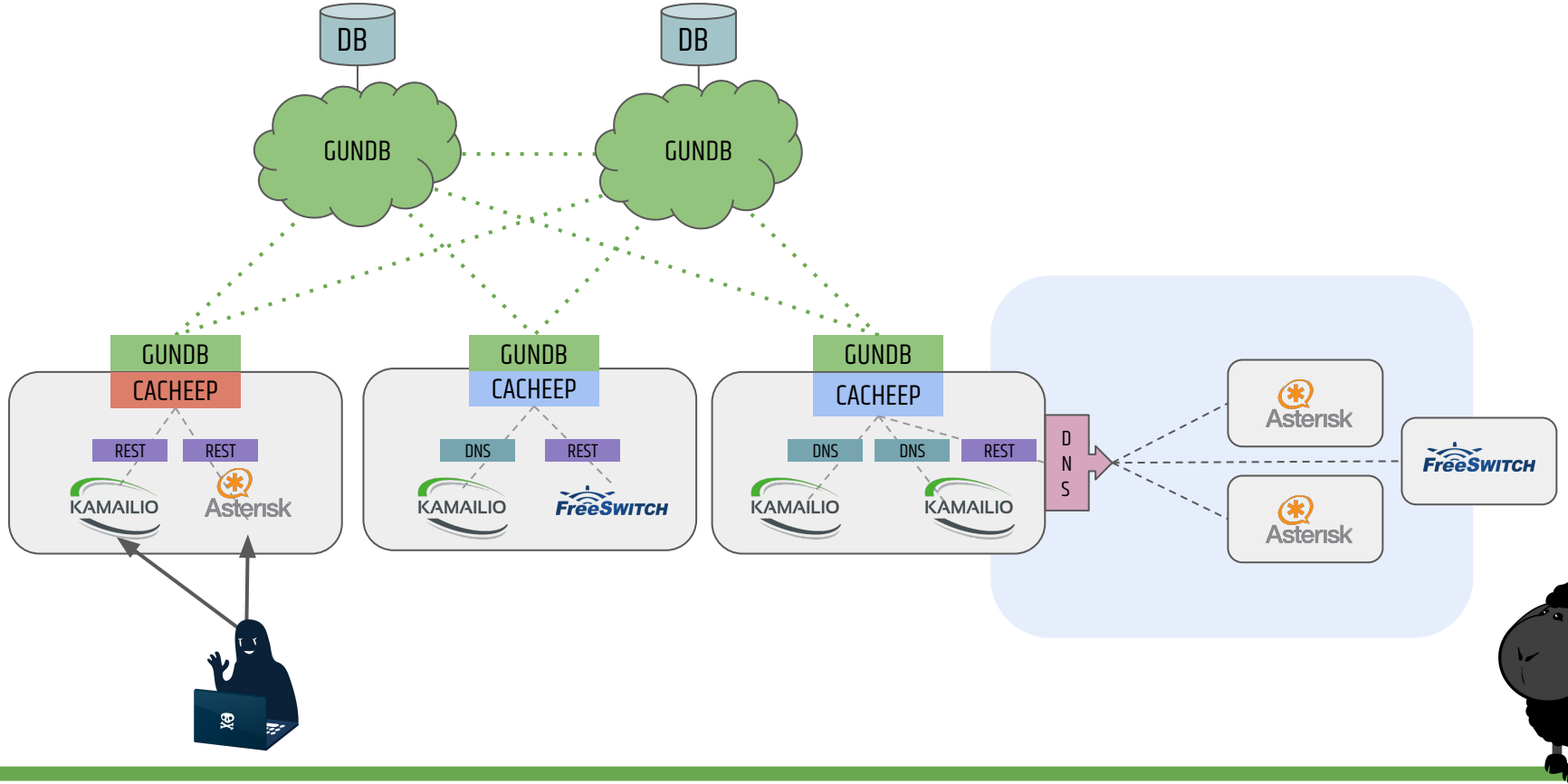
When looking for a disruptive way to approach our challenge, we stumbled upon **GUN**.

GUN is a realtime, distributed, offline-first, graph database engine able to sustain 15M+ ops/sec focusing on merge conflicts, creating an eventually consistent state across all synchronized machines. Being a graph cache-database, any data structure can be used as traditional tables with relations, document oriented trees, or tons of circular references making.

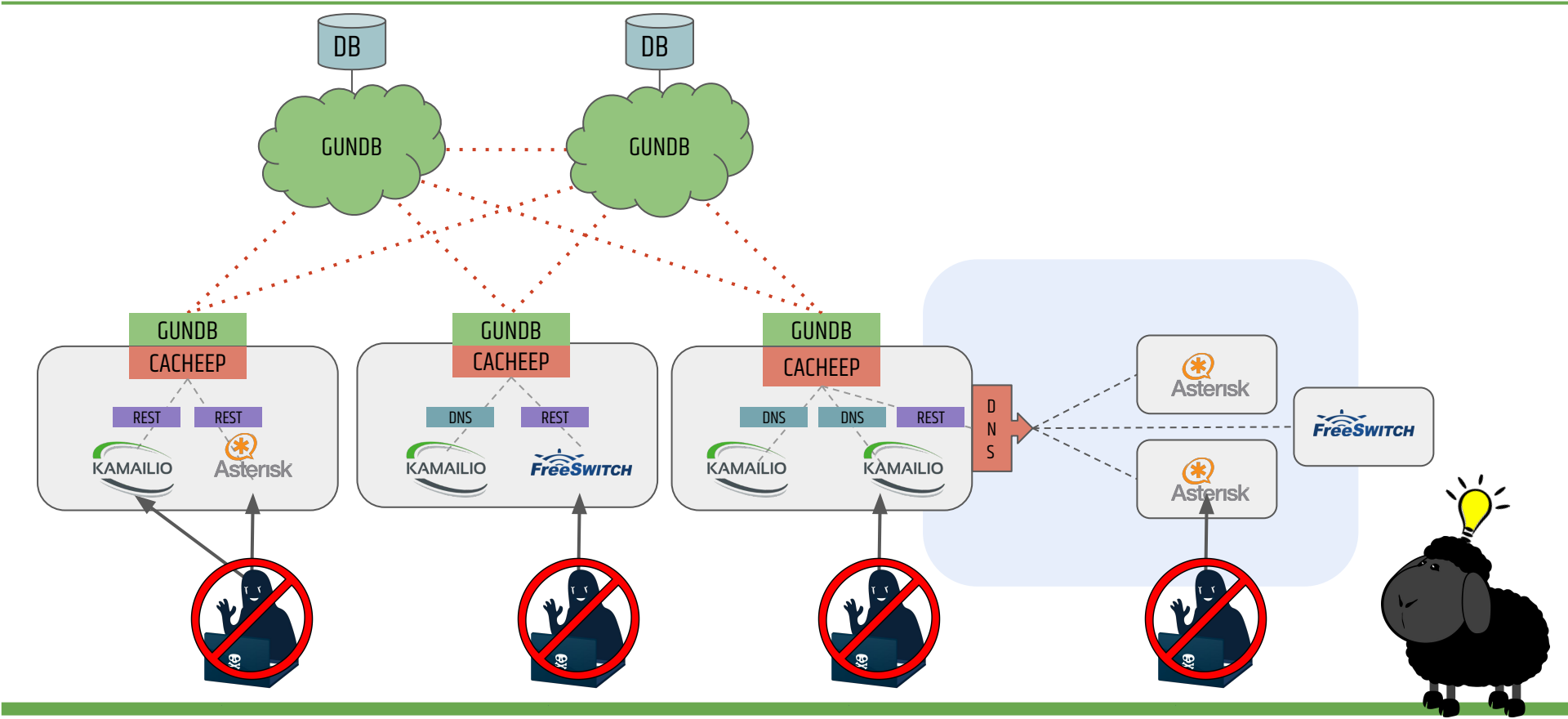
Every time a client receives data, gun makes a local copy for speed and efficiency, meaning that your most crucial data is backed up on every peer that uses it, making loss of important information *nearly impossible* as well as guaranteeing always-on access.

Soon **Cacheep** & its clients will see GunDB taking a central role - keep an eye on cross-developments at **gundb.io**

RTC Threat Intelligence Exchange



RTC Threat Intelligence Exchange

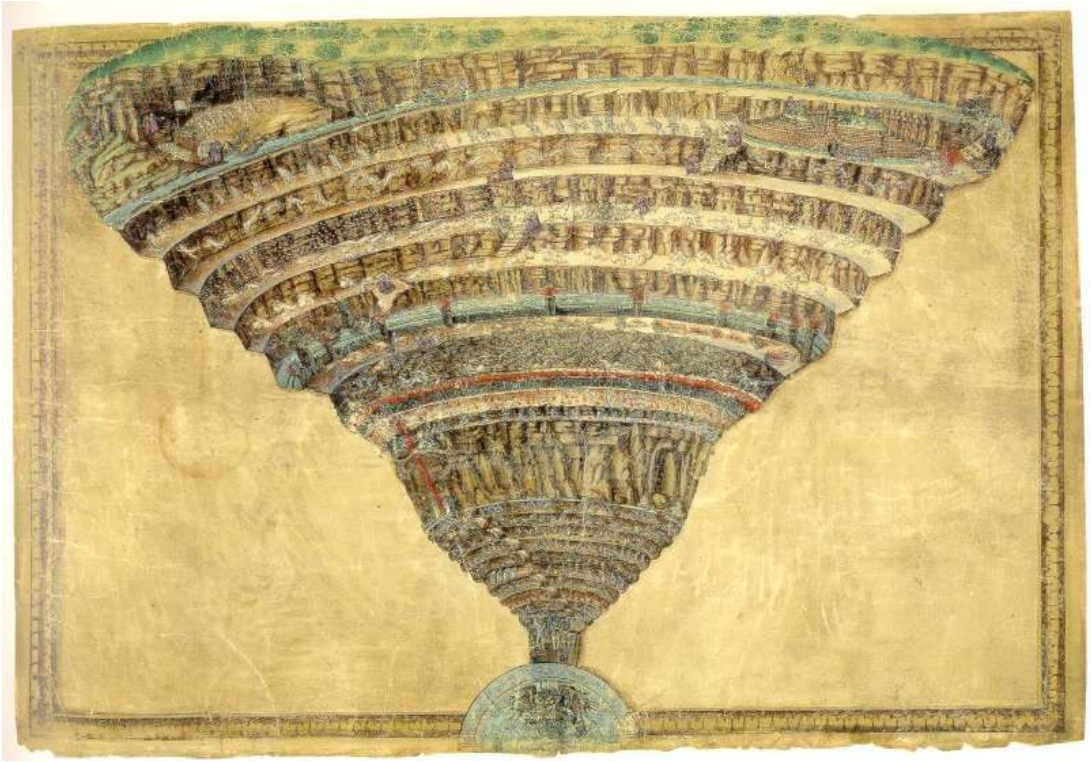


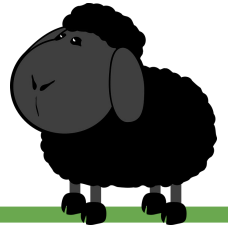
Instant Use Cases:

Cacheep can immediately be applied to develop cache-distributed solutions for the following scenarios:

- Real-Time Distributed Blacklisting
  - IP Blocks
  - E164 Destination / Prefix Blocks
- On-Demand Location & Teardown of ongoing Fraud Calls
- Number portability Cache or Redirect service
- CDR & Session Correlation
- … You Name it!

Inferno:





Inferno: Nine circles of Hell

Important to know that we should provide different blacklisted buckets/levels (BLL):

BlacklistLevel01.domain.com = Sinner, Vicious.

Reported by 3 authorization users. Maximum stay 12 hours

BlacklistLevel02.domain.com = Sinner, Vicious.

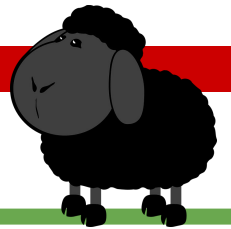
Reported by 5 authorization users and was minimum 2 times in BLL01. TTL = 1 day

BlacklistLevel02.domain.com = Sinner, Vicious.

Reported by 15 authorization users and was minimum 3 times in BLL03. TTL = 3 day

BlacklistLevel09.domain.com = Sinner, Vicious.

Permanent Lock. Only repentance.



Lock Me, Amadeus.

Here's a quick example to set self-expiring **Locks** using the generic **REST API** and instant querying from **DNS/ENUM** clients:

Block Prefix for 60 seconds

```
curl http://127.0.0.1:3000/api/set/4416/true/60000
```

Block IP for 60 seconds

```
curl http://127.0.0.1:3000/api/set/10.0.0.99/true/60000
```

ENUM e.164 Lookup

```
dig -t NAPTR 0.0.6.9.2.3.6.1.4.4.e164.arpa @127.0.0.1
```

REST e.164 Lookup

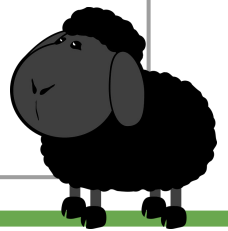
```
curl http://127.0.0.1:3000/api/get/4416
```

DNS IP Lookup

```
dig -t A 10.0.0.99.blacklist.xx.com @127.0.0.1
```

REST IP Lookup

```
curl http://127.0.0.1:3000/api/get/10.0.0.99
```



Valid locks will be instantly available for all of our **LB/Proxy/B2BUA** elements to use and query via **REST** API or using **IN-A/ENUM**. Here's a basic **Kamailio** example to check if a specific IP has been found in our local or distributed BlackList:

API Lookup

```
if (rest_get("http://127.0.0.1/api/get/$rU", "$var(reason)", "$var(ct)", "$var(rcode)")) {
    if ($var(reason) != $null) {
            xlog("Call blocked with reason: $var(reason)\n");
            send_reply("403", "Forbidden");
            exit;
    }
}
```

IN-A Lookup

```
$var(dns) = $si + ".blacklist01.botauro.com";
if( dns_int_match_ip("$var(dns)", "$si")) {
         # Blacklist Entry found!
         xlog("DNS - IN-A blacklist entry found: $si" ) ;
         route(kill) ;
}
```

Show us some Action!

Let's see how we can natively provision & integrate our prototype with the *Big Four* VoIP platforms:

- ❑ Kamailio
- ❑ OpenSIPS
- ❑ Freeswitch
- ❑ Asterisk

Kamailio has a custom DNS lookup allowing use of dnsmasq application or custom DNS root zone. We perform a lookup to our dns/cacheep if the destination is blocked. If the IP is an offender and non yet blocked we can do so using an http async query.

loadmodule "ipops.so" #DNS lookup
loadmodule "http_async_client.so" #API
loadmodule "geoip.so" #Lookup for banned IP
loadmodule "statsd.so #Statistics

```
if (is_method("REGISTER") || from_uri==myself) {
        $var(dns) = $si + ".blacklist.botauro.com";
        if (!dns_int_match_ip("$var(dns)", "$si")) {
                xlog("L_ALERT", "ip address not associated with hostname: $var(dns)\n");
                statsd_set("blacklist.nonblocked.counter,service=kamailio,region=de-west", 1);
                if($ua =~ "friendly-scanner|sipcli|VoIP SIP") {
                        xlog("L_ALERT", "Lets add the ip for block!\n");
                        $http_req(body) = "{'r_uri':'" + $rU + "', 'message':'" + $ua + "'}";
                        http_async_query("http://blacklist.xx.com:3001/api/set/$si/6000000", "HTTP_REPLY");
                        statsd_set("blacklist.sendblockrequest,service=kamailio,region=de-west", 1);
                        if(geoip_match("$si", "src")) {
                                xlog("FIRST WE BLOCKED SIP message [$si] from: $gip(src=>cc)\n");
                                statsd_incr("blacklist.blockedcountry.first,country="+$gip(src=>cc)+",service=kamailio,region=de-west");
                        }
                }
        }
        else {
                if(geoip_match("$si", "src")) {
                        xlog("REGISTER|INVITE SIP message [$si] from: $gip(src=>cc)\n");
                }
                statsd_set("blacklist.blocked.completed,service=kamailio,region=de-west", 1);
                if(geoip_match("$si", "src")) {
                        xlog("WE BLOCKED SIP message [$si] from: $gip(src=>cc)\n");
                        statsd_incr("blacklist.blockedcountry.permanent,country="+$gip(src=>cc)+",service=kamailio,region=de-west");
                }
                xlog("L_ALERT","DOMAIN BLOCKED: $var(dns)");
                sl_send_reply("503","You are blocked");
                exit;
        }
}
```

Strategy:

Unfortunately Opensips doesn't have any custom dns lookup, therefore we will use rest api client. First we check $si (source ip) and check json response to determine if destination is blocked. If so, we respond with SIP 403 and calculate stats. If not blocked, we check $ua and if it's friendly scanner we block it using the REST api

Modules:

loadmodule "rest_client.so"
loadmodule "enum.so" #In case ENUM block
loadmodule "json.so"

```
if (is_method("REGISTER | INVITE")) {
    if (rest_get("http://blacklist.xx.com:3001/api/get/$si", "$var(response)")) {
            $json(result) := $var(response);
            $var(req) = "blacklist.blocked.completed,service=opensips,region=us-west value=1 "+$Ts+"000000000";
            if (!rest_post("http://blacklist.xx.com:8186/write", "$var(req)", ,"$var(body)", "$var(ct)", "$var(rcode)")) {
                    xlog("Error code $var(rcode) in HTTP POST!\n");
            }
            if($json(result/blocked) == "true")  {
                send_reply("403","You have been banned");
                $var(req) = "blacklist.blocked.completed,service=opensips,region=us-west value=1 "+$Ts+"000000000";
                if (!rest_post("http://blacklist.xx.com:8186/write", "$var(req)", ,"$var(body)", "$var(ct)", "$var(rcode)")) {
                    xlog("Error code $var(rcode) in HTTP POST!\n");
                } exit;
            } else {
                $var(req) = "blacklist.ublocked.request,service=opensips,region=us-west value=1 "+$Ts+"000000000";
                if (!rest_post("http://blacklist.xx.com:8186/write", "$var(req)", ,"$var(body)", "$var(ct)", "$var(rcode)")) {
                    xlog("Error code $var(rcode) in HTTP POST!\n");
            }
            if($ua =~ "friendly-scanner|sipcli|VoIP SIP") {
                    $var(req) = "blacklist.block.request,service=opensips,region=us-west value=1 "+$Ts+"000000000";
                    if (!rest_post("http://blacklist.xx.com:8186/write", "$var(req)", ,"$var(body)", "$var(ct)", "$var(rcode)")) {
                                                    xlog("Error code $var(rcode) in HTTP POST!\n");
                    }
                }
            }
        };
        xlog("L_ERR", "DO LOOKUP: $si, $var(response)");
}
```

Unfortunately enum is not an option in FS.

To achieve the functionality, we use internal mod_curl and function “curl”.

We can also check UA and block it if needed.

Modules:
mod_curl

FreeSwitch

```
<extension name="announce">
  <condition field="destination_number" expression="^6000$">
    <action application="curl" data="http://blacklist.xx.com:3001/api/get/${network_addr} json" inline="true" />
    <action application="set" data="blocked=${system echo '${curl_response_data}' | jq -r '.body' | jq -r '.blocked' | tr -d '\n'}" inline="true"/>
    <condition field="blocked" expression="^true$">
          <anti-action application="respond" data="503"/>
    </condition>
    <condition field="blocked" expression="^false$">
          <action application="set" data="curl_response_data=" inline="true"/>
          <action application="info"/>
          <action application="set" data="rtcp_audio_interval_msec=5000"/>
          <action application="answer"/>
          <action application="sleep" data="2000"/>
          <action application="ivr" data="demo_ivr"/>
    </condition>
  </condition>
 </extension>
```

Same interface: CURL and check json response.

Not needed extra modules, but be sure you have curl installed.

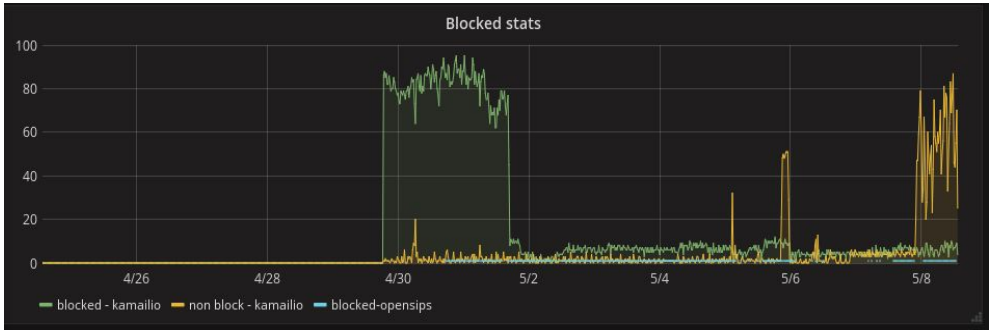
Asterisk

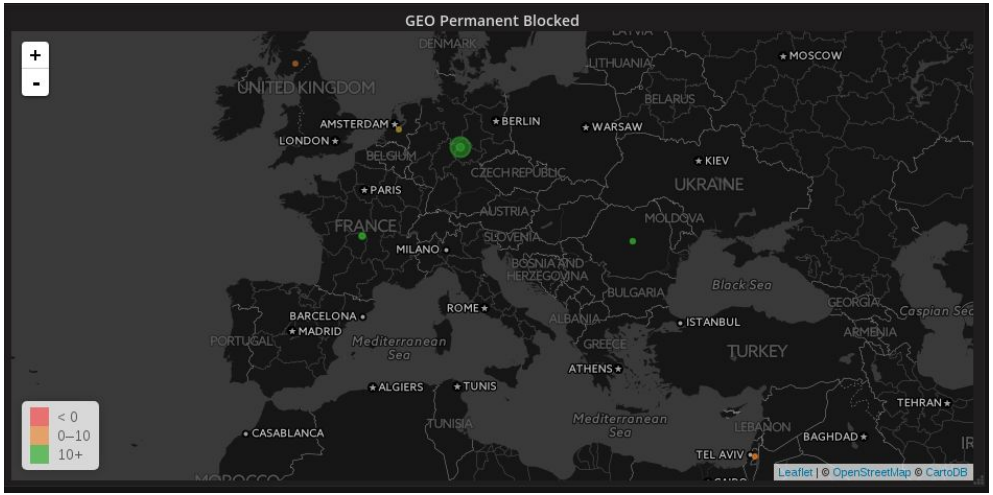
```
exten => _X.,1,Set(CURL_RESULT=${CURL(blacklist.xx.com:3001/api/get/${SIPCHANINFO(peerip)})})
same => n,Set(result=${JSONELEMENT(CURL_RESULT,blocked)})
same => n,GotoIf($["${result}" = "1"]?block:process)
same => n(block),Hangup
same => n(process),Answer
same => n,Hangup
```

Statistics!

Nothing makes sense without statistics and detailed metrics to understand how our systems operate and react.

...





Killing Fraud. No Mercy.

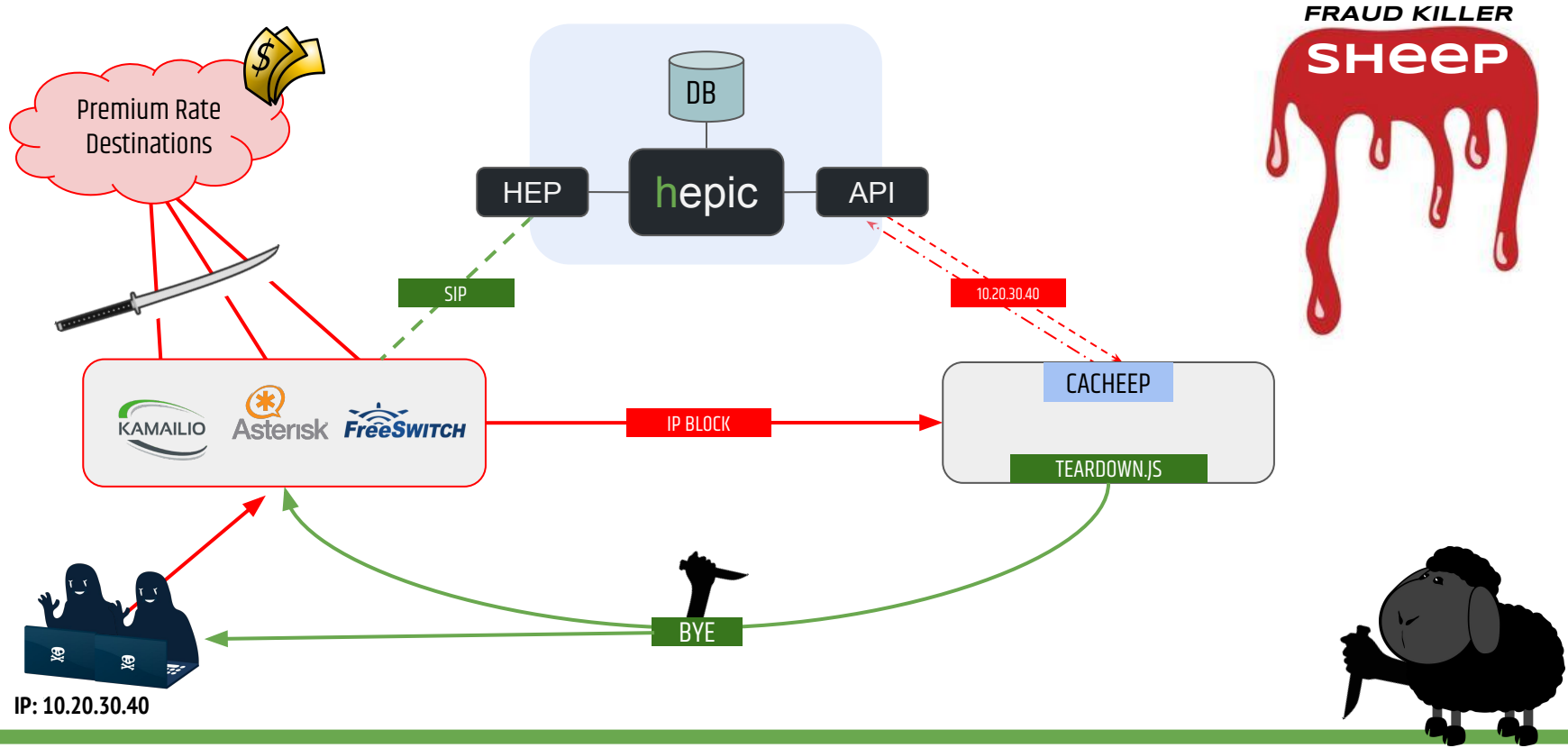
Let's say one of our federated nodes detects and Bans an Attacker - any new call will be instantly blocked - that's great!
But what happens to any established and ongoing fraud call$?

Quite too often, Nothing - They keep *vacuuming* money!

That's sad and potentially very, very expensive, too.

This is where our **Teardown** module comes into play, inspired by the same tools our attackers use against our users.
SIP is has a few weaknesses, and if criminals take advantage of them, so should we for defense! Say good **BYE** to Fraud!

By leveraging **HEPIC** integration, **Cacheep** can query for any established calls by an **IP** or **User/Destination**, rebuild the key sessions parameters and forge a perfectly valid **BYE Teardown** message for both directions, stopping the **leak** right there.


FRAUD KILLER
SHeeP
Premium Rate Destinations
DB
HEP
hepic
API
SIP
10.20.30.40
KAMAILIO
Asterisk
FreeSWITCH
IP BLOCK
CACHEEP
TEARDOWN.JS
BYE
IP: 10.20.30.40

Let's Make it Happen.

The key concept is already welcomed by the leading Projects in our ecosystem and the technical proposals are already gaining some interesting concepts, but in order for this to succeed, We need **YOU**, too!

Developer?	We need your expertise and ideas to make this baby fly high.
Architect?	We need your help to scale this idea beyond the horizon
Security Expert?	Let's make this unbreakable and keep it unpoisoned
Operator?	We need your support and backing to empower this ecosystem
OSS Politician?	Help us by making this initiative known to potential Angel backers.
Not a Technical guy?	We'll need help with arranging, coordinating, documenting.

Interested? Contact us today: (needs mailing-list or gitter)

Thank you!