Kamailio® SIP security

Henning Westerholt

Kamailio World

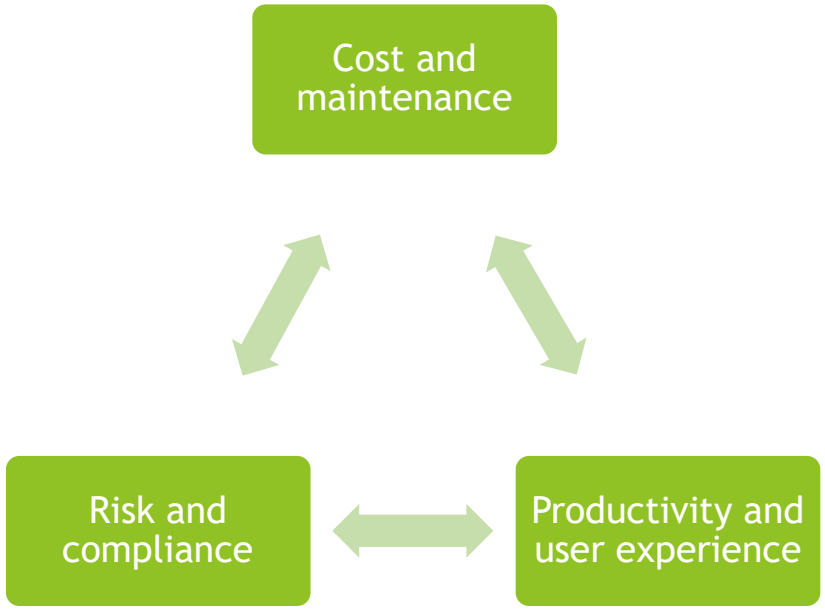
May 2019 - Berlin

Agenda

- About me
- The basics
- Security by obscurity
- Check message validity
- DOS infrastructure protection
- Detecting malicious behaviour
- Improvement for authentication
- Operational notes
- Contact

About me

- Henning Westerholt
- With Kamailio project since 2007
- Core developer of the Kamailio® project, member of management board
  - Core, database work and different other modules
  - Administration, code quality, security testing, quality assurance
  - Kamailio fuzzing project
- Company started in 2018
  - Consulting for Kamailio and Voice over IP services
  - Services and development, Workshops and trainings



- Security is always a trade-off, you need to weight it against:
  - Risk
  - Convenience
  - User friendliness
  - Implementation effort
  - Maintenance costs and complexity
  - Support costs and customer skill level

The basics

Keep your system up to date

Use a support distribution release
Actually apply the updates in time
(at least monthly)

Protect your system

Only login for necessary users, SSH keys
Provisioning only necessary services
Bind databases and other APIs only to localhost or private network
Use fail2ban to protect your ssh login

Keep Kamailio up to date

The last two stables releases are supported (e.g. 5.1 and 5.2)
End of maintenance will be announced
Minor releases are provided every
1-2 months

Security by obscurity (1/2)

- There are different opinions on this, of course – nevertheless it can be useful against unskilled attackers (script kiddies)
- System setup
  - Use non-standard port for SSH or other management services
  - Don’t publish all internal machines in your external DNS
- Kamailio server
  - Don’t announce exact Kamailio version, or even no details at all

```
server_header="Server: Kamailio" or disable completely with
"server_signature=no"

user_agent_header="User-Agent: Kamailio"

sip_warning=0
```

  - Disable debug feedback (e.g. *"Noisy feedback tells: pid=2489 req_src_ip=X.X.X.X req_src_port=2681 in_uri=sip: a@Y.Y.Y.Y out_uri=sip: b@Z.Z.Z.Z via_cnt==1)*

Security by obscurity (2/2)

- Use topoh to hide internal network details like PSTN gateways or carriers
  - Topoh module supports two modes: header storage and dialog variable storage
  - Dialog variable storage is more secure, as data will be removed completely
  - Header storage needs to be protected with a random mask_key parameter
- The topoh module transparently rewrites Via and Contact headers
- Important topoh parameter

```
modparam("topoh", "mask_key", "my-random-key")
modparam("topoh", "mask_ip", "10.0.0.1") # unused IP
modparam("topoh", "mask_callid", 1)
```

- Check messages for additional protection

```
modparam("topoh", "sanity_checks", 1)
modparam("topoh", "uri_prefix_checks", 1)
```

Use fail2ban to protect your server

- Use fail2ban to protect your server on a network level
  - /etc/fail2ban/filter.d/kamailio.conf:
  - [Definition]
  - failregex = Blocking traffic from <HOST>
  - /etc/fail2ban/jail.conf:
  - [kamailio-iptables]
  - enabled = true
  - filter   = kamailio
  - action   = iptables-allports[name=KAMAILIO, protocol=all]
  - logpath = /var/log/kamailio.log # use your kamailio log path
  - maxretry = 10
  - bantime = 1800
  - kamailio.cfg: xlog("Blocking traffic from $si\n");

Check message validity (1/2)

- The sanity module can be used also from topos (more modules tbd.)
- You should use the sanity module for all incoming messages and also replies
  - Check for headers and URI content

```
if(!sanity_check("17895", "7")) {
    xlog("Malformed SIP request from $si:$sp\n");
    exit; }
```

- Drop messages from script kiddie scanner and testing tools

```
if($ua =~ "friendly-scanner|sipcli|sipvicious|VaxSIPUserAgent") {
    xlog("L_INFO", "scan from $si:$sp - $ua\n");
    exit; }
```

- Drop or reply with 200 OK – adapt to your general security policy

Check message validity (2/2)

- Drop SQL injection attacks (adaption to your setup necessary)

```
if($au =~ "(\=)|(\-\-)|(')|(\#)|(\%27)|(\%24)") {
    xlog("L_INFO","injection from $si:$sp - $au\n");
    exit; }
```

- Check message IPs (additional to firewalling)
  - You can use e.g. the permissions module
  - If you have a small number of IPs you can also just use script logic with $si
- Check against spoofing attacks
  - Compare incoming IP address $si with Record-Route, Contact and/or Via header

```
if($(hdr(Record-Route)[0]{nameaddr.uri}) != $si) {
    xlog("L_INFO", "Spoofing attack from $si, blocking");
    exit; }
```

DOS infrastructure protection (1/2)

- Consider using the destination blacklist support, to prevent expensive sending or transaction logic for repeatedly failing destinations

```
use_dst_blacklist=on
```

  - Have a look to the core cookbook for details on GC collection time (link)

- Use the ratelimit module to enforce an upper limit for certain message types in your infrastructure, to protect application servers or gateways

```
modparam("ratelimit", "queue", "0:*")
modparam("ratelimit", "pipe", "0:TAILDROP:200") # 200/timer
if (is_method("INVITE|REGISTER|SUBSCRIBE") {
    if (!rl_check()) {
        append_to_reply("Retry-After: 5\r\n");
        sl_send_reply("503","Limiting");
        exit; } }
```

- Consider to randomizing the Retry-After header to prevent Client synchronization

DOS infrastructure protection (2/2)

- Use the pike and htable module for IP based blocking of incoming requests
  - Pike module uses a subnet approach for calculating the blocking, i.e. if several requests come from one network subsequent IPs will be blocked faster (link)
  - Taken from the default cfg

```
modparam("htable", "htable", "ipban=>size=8;autoexpire=300;")
if(src_ip!=myself) { # add also trusted peers here
    if($sht(ipban=>$si)!=$null) { # ip is already blocked
        xdbg("blocked IP - $rm from $fu (IP:$si:$sp)\n");
        exit; }
    if (!pike_check_req()) {
        xlog("L_ALERT", "blocking $rm from $fu (IP:$si:$sp)\n");
        $sht(ipban=>$si) = 1;
        exit; }
}
```

Blocking password bruteforcing (1/3)

- Slower attacks are harder to detect, as you don't want to block valid users
- But you can protect against slow password brute-forcing with htable
  - `modparam("htable", "htable", "a=>size=8;autoexpire=920;")`

```
if(is_present_hf("Authorization")) {
    if($sht(a=>$au::auth_count)==3) {
        $var(exp) = $Ts - 900;
        if($sht(a=>$au::last_auth) > $var(exp)) {
            send_reply("403", "Try later"); # local policy
            exit;
        } else {
            $sht(a=>$au::auth_count) = 0; }
        }
    }
```

Blocking password bruteforcing (2/3)

```
if(!www_authenticate("$td", "subscriber")) {
    switch ($retcode) {
        case -1: sl_send_reply("403", "Forbidden"); exit;
        case -2:
            if($sht(a=>$au::auth_count) == null)
                $sht(a=>$au::auth_count) = 0;
            $sht(a=>$au::auth_count) = $sht(a=>$au::auth_count) + 1;
            if($sht(a=>$au::auth_count) == 3)
                xlog("auth failed 3rd time - src ip: $si\n");
            $sht(a=>$au::last_auth) = $Ts;
        break;
    }
    www_challenge("$td", "0"); exit;
}
```

Blocking password bruteforcing (3/3)

- Reset count after successful authentication

```
$sht(a=>$au::auth_count) = 0;
```

- Challenge request without Authorization header

```
        } else {
            www_challenge("$td","0");
            exit;
        }
    }
```

Blocking parallel call fraud

- Use the dialog module to limit parallel calls (one instance)

```
modparam("dialog", "profiles_with_value", "concurrent_calls")
if (!dlg_isflagset("1")) { # $td - gateways, $fu - user
    if (get_profile_size("concurrent_calls", "$td", "$avp(calls)")) {
        if ($avp(calls) >= 5)) {
            xlog("L_INFO", "Concurrent calls $td at limit");
            send_reply("503", "Calls limit reached"); exit;
        } else {
            dlg_manage(); dlg_setflag("1");
            set_dlg_profile("concurrent_calls", "$td");
        }
    }
}
```

- Manage multiple instances with remote dialog profiles (link) or by using a DB

Flexible blocking modules

- Many modules offer blacklisting functionality (as interface to a database and/or with caching support for better performance)
- Nowadays this functionality can be implemented with sqlops and htable
- Nevertheless some of the common modules are listed with some examples
- The userblacklist module can be used to block certain destinations

```
check_user_blacklist(user, domain, number, table) # user DB
check_blacklist(table) # global list, with in-memory caching
```

- The secfilter module can be used to black certain user agent or countries

```
secf_check_ua() # for user agents
secf_check_country($gip(src=>cc) # for countries, with geoip
```

- Have a look to the module documentation for more inspiration

Notes on user authentification

- Standard SIP digest authentication unfortunately still uses MD5 as hashing algorithm
- MD5 is broken since a long time, but unfortunately is widely used
- HTTP Digest Authentication already includes SHA256 and SHA512
- Draft IETF “sipcore-digest-scheme” updates (link) also the SIP authentication standard
- The auth module already supports SHA256
- So give it a try with your user agents, looking forward to your feedback

Improve user authentication

- You can improve the auth module security by configuring it better (and of course using TLS)
- The following settings should be supported from most user agents

```
modparam("auth_db", "calculate_ha1", 1)
modparam("auth", "nonce_count", 1)
modparam("auth", "qop", "auth")
modparam("auth", "nonce_expire", 60)
modparam("auth", "nonce_auth_max_drift", 2)
```

- This sections can break authentication, extensive tests recommended

```
# hash the Request-URI, Call-ID, and source IP into the nonce
# Break user agents that change Call-ID for every REGISTER message
modparam("auth", "auth_checks_register", 11)
# hash the Request-URI and source IP for dialog forming requests
modparam("auth", "auth_checks_no_dlg", 9)
# hash source IP, Request-URI, Call-ID and From tag
modparam("auth", "auth_checks_in_dlg", 15)
```

Operational notes

Don’t forget the basics: keep your system up to date, protect your system, use a current Kamailio version

Do not change too many things at once, keep track of your changes

Add logging information when a check triggers and something is blocked

Keep in mind that certain user agents might not behave correctly

Thank you for your attention

- Contact:
  - Henning Westerholt
  - mail@skalatan.de
  - https://skalatan.de/services