Diseño e implementación: Prometheus node exporter en C para OpenBSD

Abel Camarillo <acamari@verlet.org>

28 de junio de 2019

Agenda

- ¿Quién soy?
- ¿Qué es?
- ¿Por qué?
- ¿Cómo?

¿Quién soy?

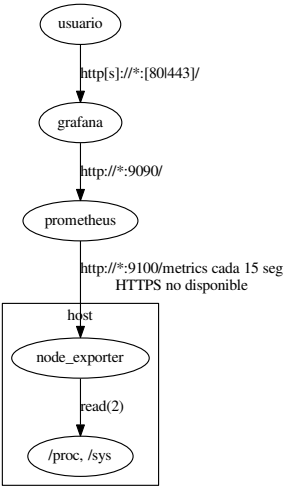
- Desarrollador de software desde el 2008 - OpenBSD, perl, C, sh, js
- Lead developer en Neuroservices Communications durante 6 años.
- Desarrollador freelance desde el 2015 - Verlet.
- Maintainer de 21 paquetes en el árbol oficial de OpenBSD - http://openports.se/bbmaint.php?\maint=acamari@verlet.org
- Interés en UNIX, poesía, ~arte~, cocina, etc...

¿Y qué carajo le importa al agua? - 2011

¿Qué es?

- ¿Qué es Prometheus?
- ¿Qué es OpenBSD?
- ¿Qué es C?
- ¿Qué es un exporter?
- ¿Qué es el node_exporter?

Arquitectura original



Arquitectura prometheus (original)

¿Por qué?

Quiero monitorear mi router.

- Ubiquiti EdgeRouter-Lite (dual octeon@500mhz)
- OpenBSD-6.5/mips64 (dual-endianess, OpenBSD escogió BE)
- go sólo disponible en i386 y amd64
- ¿Por qué no portar go?: go-bootstrap
- Aunque hubiera go:
  - node_exporter incompleto en *BSD (/proc, /sys)
- Alternativas:
  - prometheus_sysctl_exporter (FreeBSD)
  - Incompatible con node_exporter

¿Por qué?

Sí, pero, ¿en C?

- No hay /proc, /sys.
- Disponibilidad de sysctl(3)
- Acceso a unveil(2), pledge(2)
- Sí hay perl en OpenBSD base, pero no BSD::Sysctl. Además third-party. Sino: perlxs, (un)pack, etc...
- No hay python en OpenBSD base. Acceso sysctl indefinido.
- ¿nodejs?: jajajaja

¿Cómo?

Diseño

- Formato de intercomunicación
- Seguridad
- Arquitectura
- Integración
- Simplicidad
- Interfaz

Formato de intercomunicación

V4. Definido en github de prometheus (\ agregados para legibilidad):

https://github.com/prometheus/docs/blob/master/content/docs/instrumenting/exposition_formats.md

Ejemplo:

```
$ curl http://*:9100/metrics;
...
# HELP node_forks_total Total number of forks.
# TYPE node_forks_total counter
node_forks_total 1.8757377e+07
# HELP node_load1 1m load average.
# TYPE node_load1 gauge
node_load1 1
# HELP node_load15 15m load average.
# TYPE node_load15 gauge
node_load15 1.37
# HELP node_load5 5m load average.
# TYPE node_load5 gauge
```

```
node_load5 1.15
# HELP node_uname_info Labeled system information as provided by   \
        the uname
system call.
# TYPE node_uname_info gauge
node_uname_info{domainname="(none)",machine="x86_64",nodename="db4"\
    ,release="4.15.12-x86_64-linode105",sysname="Linux",\
    version="#1 SMP Thu Mar 22 02:13:40 UTC 2018"} 1
# HELP process_cpu_seconds_total Total user and system CPU time \
    spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 2111.27
# HELP node_cpu_seconds_total Seconds the cpus spent in each mode.
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="0",mode="idle"} 2.980243359e+07
node_cpu_seconds_total{cpu="0",mode="iowait"} 8497.06
node_cpu_seconds_total{cpu="0",mode="irq"} 0
node_cpu_seconds_total{cpu="0",mode="nice"} 20709.44
```

Formato de intercomunicación

- Una métrica puede tener varios valores si hay etiquetas
- Las etiquetas pueden tener longitud arbitraria
- Etiquetas pueden variar entre ejecuciones: fs nuevos, tarjetas de red o IPs entran y salen, etc
- Se observaron crasheos en las condiciones anteriores

Seguridad

node_exporter:

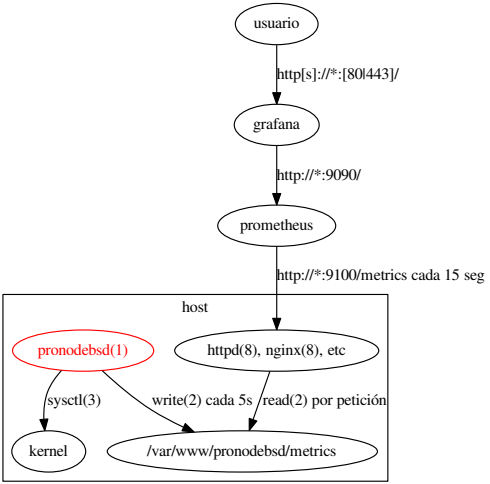
- Amplia superficie de ataque: sockets (DOS), http parsing
- Mucha gente lo corre como root
- Sin soporte HTTPS

Seguridad

pronodebsd:

- Sin web server: desplaza superficie de ataque a otra capa
- No analiza entradas externas (aparte de argumentos de inicio y salida de sysctl(3))
- ¿Si sysctl regresa basura? Hay problemas más graves
- `unveil("/var/www/pronodebsd/", "rw"); unveil(NULL, NULL);`
- Usuario sin privilegios

Arquitectura pronodebsd



Arquitectura prometheus (pronodebsd)

Integración

- Logueo tradicional: syslog, newsyslog, logrotate, etc
- Manejo de señales: SIGHUP, SIGINT, etc
- Manejo atómico de archivos:
  - No queremos escribir archivo metrics a medias, porque en el resto del stack no se podría distinguir la falta de métrica vs archivo a medias
  - Usar archivo temporal /var/www/pronodebsd/.metrics
  - `rename("./.metrics", "./metrics")`

Simplicidad

- No hay necesidad de recalcular stats <5seg
- Es el tiempo de refresco de muchos parámetros de kernel: loadavg, temperatura, etc
- Facilidad para añadir collectors
- Manejo de memoria de heap fuera de los collectors, prevenir leaks. No regresar memoria estática, ni global, ni malloc(3) (sin free(3) léxico) dentro de collectors

Interfaz

```
pronodebsd [–d dir] [–s secs] [–f]
        –d directory (/var/www/pronodebsd)
        –s seconds (5)
        –f foreground, ignore –s and –d and print once to stdout
```

¿Cómo?

Implementación

- Estructuras de datos
- Collectors
- Status

Implementación

pronodebsd.c:

```
struct mresult {
        int     labelsz;        /* size allocated for labels */
        int     nlabelsz;       /* size should be allocated for labels */
        char    *label;         /* one or more labels */
        double  value;
};
```

Implementación

pronodebsd.c:

```
enum METRIC_TYPES { MTYPE_UNTYPED = 0, MTYPE_COUNTER, MTYPE_GAUGE,
                        MTYPE_HISTOGRAM, MTYPE_SUMMARY };

static struct metrics_t {
        char    *name;
        char    *help;
        enum METRIC_TYPES       type;
        /* use only one of collector or collectorv */
        int     (*collector)(double *, char **);
        int     (*collectorv)(struct mresult *, int, char **, int *);
        int     nelem; /* number of elements the collector returned */
        /* storage for collectors that return complex results */
        struct mresult  *mres;
};
```

Implementación

pronodebsd.c:

```
static struct metrics_t metrics[] = {
        ...
        { ”node_intr_total”, ”Total number of interrupts serviced.”,
            MTYPE_GAUGE, intr_collector, NULL, 0, NULL },
        { ”node_uname_info”, ”Labeled system information as ”
             ”provided by the uname system call.”, MTYPE_GAUGE, NULL,
             uname_collector, 0, NULL },
        { NULL, NULL, 0, NULL, NULL, 0, NULL }
};
```

Implementación

sysctl.c:

```
#include <sys/types.h>
#include <sys/sysctl.h>
#include <sys/vmmeter.h>
#include <uvm/uvmexp.h>

#include <stdlib.h>
#include <string.h>
#include <errno.h>

#include ”sysctl.h”

static int
uvmexp_collector(struct uvmexp *uvmexp, char **err)
{
        size_t  sz = sizeof *uvmexp;
        int     mib[] = { CTL_VM, VM_UVMEXP };
```

```
        if (sysctl(mib, sizeof mib / sizeof mib[0], uvmexp, &sz,
                   NULL, 0) == -1) {
                *err = strerror(errno);
                return -1;
        }
        return 0;
}

int
intr_collector(double *result, char **err)
{
        struct uvmexp   uvmexp;

        if (uvmexp_collector(&uvmexp, err) == -1) {
                return -1;
        }

        *result = uvmexp.intrs;
        return 0;
}
```

Implementación

sysctl.c:

```
#include <sys/utsname.h>
static int
uname_collector(struct mresult *mres, int nelem, char **err,
        int *newnelem)
{
        struct utsname  un;
        char buf[1024] = "";
        /* max number of elements this func will return */
        const int maxelem = 1;
        int c;

        *newnelem = maxelem;
        if (nelem < maxelem) {
                return 0;
        }

        if (uname(&un) == -1) {
```

```
        *err = strerror(errno);
        return -1;
}

c = snprintf(buf, sizeof buf,
                "domainname=\"(none)\","
                "machine=\"%s\","
                "nodename=\"%s\","
                "release=\"%s\","
                "version=\"%s\"",
                un.machine,
                un.nodename,
                un.release,
                un.version);
if (c < 0) {
        *err = "snprintf";
        return -1;
}

mres->nlabelsz = c + 1; /* add \0 */
```

```
        if (mres->labelsz < mres->nlabelsz) { /* not enough space */
                return 0;
        }

        mres->labelsz = mres->nlabelsz;
        mres->value = 1;
        strlcpy(mres->label, buf, mres->labelsz);
        return 1;
}
```

Status

Falta:

- Escribir archivo real
- Loop de actualización
- Muchísimas métricas, es trabajo simple pero tedioso
- Integración con «init» de OpenBSD rc(8)
- Hacer port, para poder hacer pkg_add pronodebsd
- manpage, README de instalación

¿Preguntas?