



cesnet
"...."

NERD

ÚVOD PRO VÝVOJÁŘE

Václav Bartoš

- Co je to NERD + ukázka
- Princip fungování, architektura, UpdateManager a moduly, konfigurace
- Uspořádání repozitáře
- Jak vytvořit vlastní modul
- Aktuální moduly, datový model
- Nová architektura
- Plány do budoucna

Co je to NERD?

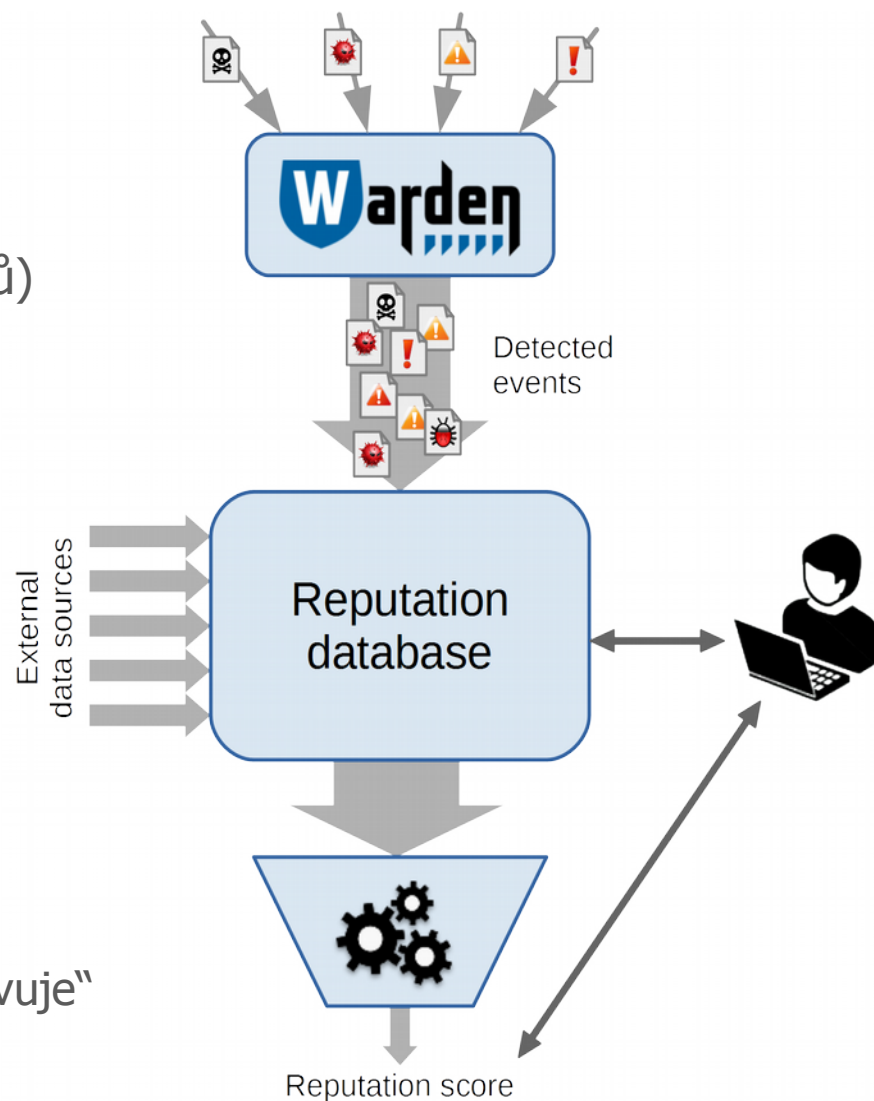
■ Databáze síťových entit

- **IP adresy**, sítě, domény, ...
- Seznam známých zdrojů škodlivých aktivit na internetu a všeho, co o nich víme
- Mnohem víc, než jen blacklist

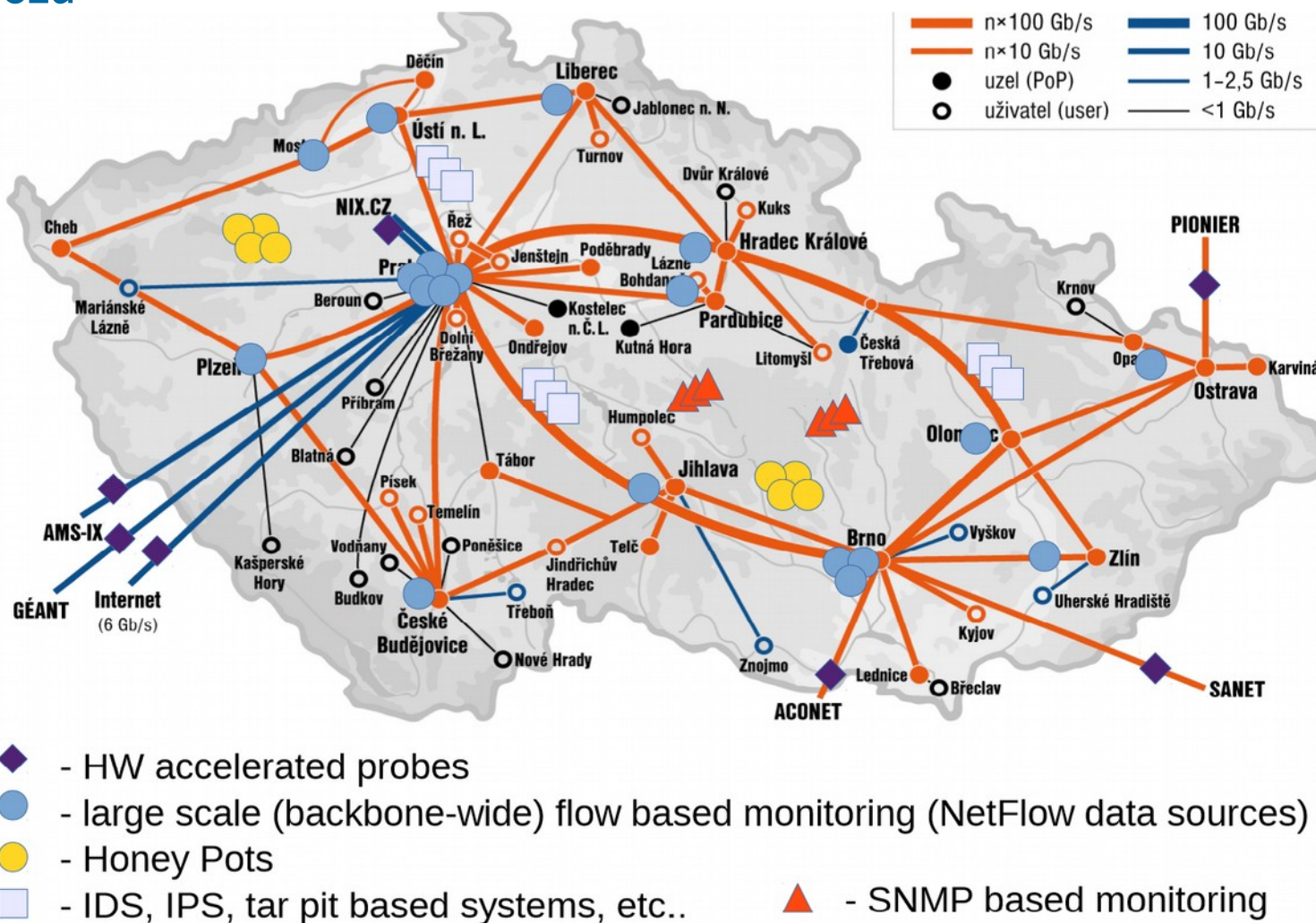


■ Základní princip fungování:

- Přijímá hlášení o bezpečnostních událostech z Wardenu (příp. i z jiných obdobných systémů)
 - Agregace podle zdrojové adresy
- Obohacení o další data z externích zdrojů
 - Hostname, ASN, geolokace, ...
 - Přítomnost na blacklistech
 - Open[DNS,NTP,...] resolvers
 - TOR exit nodes
 - ...
- Shrnutí všech informací do „reputation score“
 - Ohodnocení „jak velkou hrozbu entita představuje“

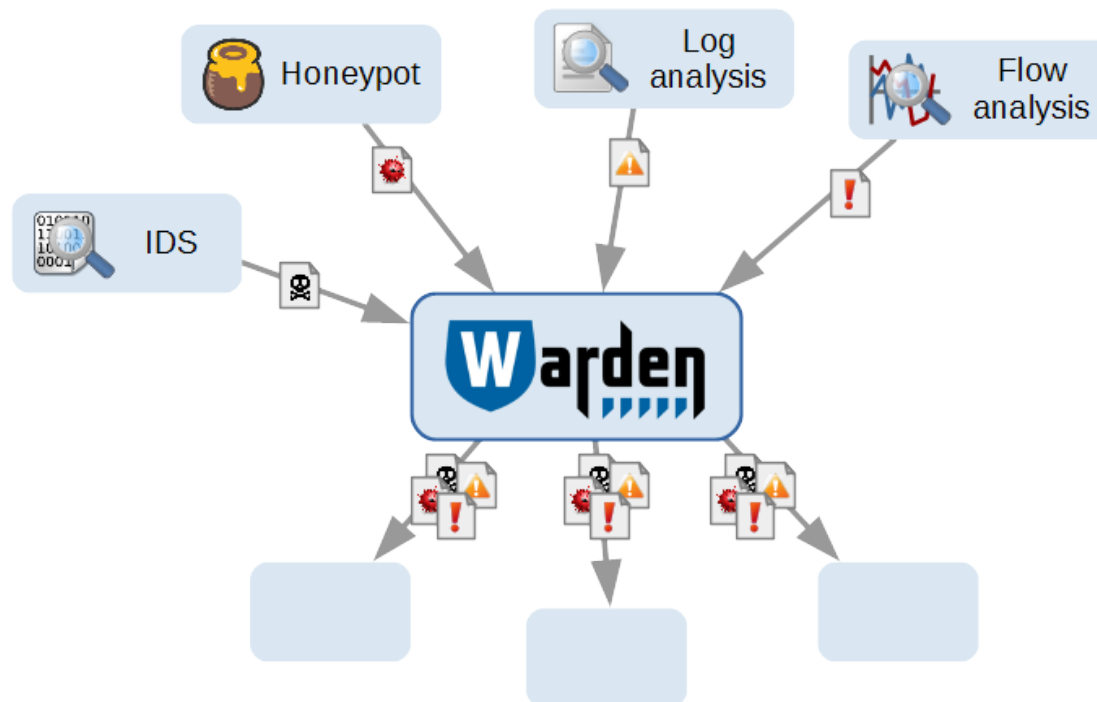


- V síti CESNET a na univerzitách je množství systémů pro detekci škodlivého provozu



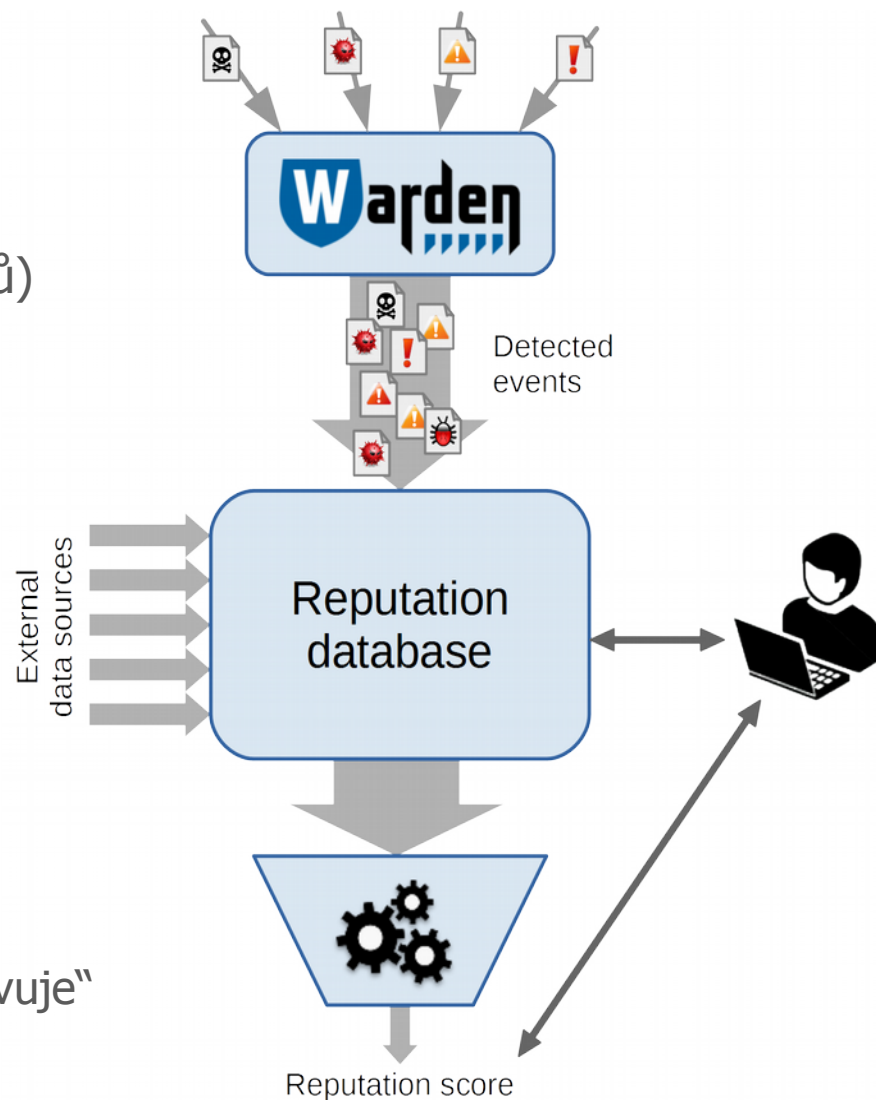
■ Warden – systém pro sdílení „alertů“ v CESNETu

- Detektory posílají zprávy o detekovaných událostech na centrální server, ten je distribuuje přijímajícím klientům
- Komunitní přístup: každý, kdo provozuje detektor a sdílí data, má přístup k datům všech ostatních
- Jednotný datový formát (IDEA, <https://idea.cesnet.cz/>)



■ Základní princip fungování:

- Přijímá hlášení o bezpečnostních událostech z Wardenu (příp. i z jiných obdobných systémů)
 - Agregace podle zdrojové adresy
- Obohacení o další data z externích zdrojů
 - Hostname, ASN, geolokace, ...
 - Přítomnost na blacklistech
 - Open[DNS,NTP,...] resolvers
 - TOR exit nodes
 - ...
- Shrnutí všech informací do „reputation score“
 - Ohodnocení „jak velkou hrozbu entita představuje“



■ Uživatelé

- CSIRT týmy
- Síťoví administrátoři
- Výzkumníci v oblasti síťové bezpečnosti
- (částečně i veřejnost)

■ Možnosti použití

- Upozornění na zlé adresy ve vlastní síti
- Řešení a vyšetřování bezpečnostních incidentů
- Generování blacklistů (blokování provozu z hodně škodlivých adres)
- Statistiky, výzkum
- Cokoliv dalšího, s čím někdo přijde :)

■ Přístup k datům

- Jednoduchý web + REST API

- Hlavní web:
 - <https://nerd.cesnet.cz/>
- Zdrojáky + wiki:
 - <https://github.com/CESNET/NERD>

NERD [List of IPs](#) | [IP detail](#)

Logged in: washek (Václav Bartoš) • [Log out](#)

Known IP addresses

IP prefix: & Hostname suffix: & Country code: & ASN:

Event category: & Node: & Tag: Min tag confidence: 0.5

Sort by: Time of last event ☐ Ascending

Max number of addresses: 20

Results (220)

IP address	Hostname	ASN	Country	Events	Rep.	Other properties	Time added	Last event	Links
191.107.235		??	US	49635 7 1	0.969	uceprotect blocklist_de-portflood	2017-10-06 12:01:07	2017-12-14 15:03:50	🔗 🌐
66.249.138		AS10439	US				2016-07-11 17:56:28	2017-12-14 15:03:49	🔗 🌐
193.50.38		AS8267	PL				2016-09-10 22:13:07	2017-12-14 15:03:49	🔗 🌐
45.55.8	...k.com	AS63949	US				2016-10-12 08:29:03	2017-12-14 15:03:49	🔗 🌐
80.6.10	...n.io	AS29073	NL			blocklist_de-mail	2017-03-20 20:06:04	2017-12-14 15:03:49	🔗 🌐
209.85.136		AS10439	US				2017-06-07 17:52:03	2017-12-14 15:03:49	🔗 🌐
191.107.245		??	US				2017-09-27 15:46:16	2017-12-14 15:03:49	🔗 🌐
202.110.110	ftth.katch.ne.jp	AS59108	JP	3904 4 1	0.325	uceprotect spamhaus-xbl-cbl	2017-10-29 00:46:16	2017-12-14 15:03:49	🔗 🌐
180.150.150		AS4812	CN	6867 5 2	0.911	spamhaus-xbl-cbl blocklist_de-ssh uceprotect bruteforceblocker	2017-11-23 04:52:35	2017-12-14 15:03:49	🔗 🌐
163.198.198	ev.poneytelecom.eu	781	GB	2 1	0.193		2017-12-13 06:02:53	2017-12-14 15:03:49	🔗 🌐
139.76.76	pip.net	AS63949	US	56113 7 1	0.899	uceprotect	2017-07-24 05:17:33	2017-12-14 15:03:48	🔗 🌐
117.195.195			CN	668 3 1	0.225	uceprotect	2017-12-13 13:42:52	2017-12-14 15:03:48	🔗 🌐
103.15.15		AS132116	IN	3569 5 1	0.715	Scanner spamhaus-pbl spamhaus-xbl-cbl	2017-03-03 22:04:09	2017-12-14 15:03:38	🔗 🌐
181.239.239		??	US	20048 7 1	0.656	uceprotect blocklist_de-portflood	2017-09-25 20:05:25	2017-12-14 15:03:34	🔗 🌐
117.19.19		AS4134	CN	8448 1 1	0.500	Scanner	2017-06-19 19:16:45	2017-12-14 15:03:33	🔗 🌐
124.199.199		AS4837	CN	8545 1 1	0.500	Scanner	2017-06-19 19:17:25	2017-12-14 15:03:33	🔗 🌐
112.119.119		AS56041	CN	8580 1 1	0.500	Scanner	2017-06-19 19:16:31	2017-12-14 15:03:32	🔗 🌐
202.167.167		AS4637	HK	8471 1 1	0.500	Scanner	2017-06-21 13:19:16	2017-12-14 15:03:32	🔗 🌐
141.210.210	eeecs.umich.edu	AS36375	US	12742 4 1	0.875	Scanner	2016-07-11 21:23:54	2017-12-14 15:03:29	🔗 🌐
71.6.6.6	...io	AS10439	US	54809 9 3	0.934	Scanner Research scanner uceprotect	2016-07-11 17:56:31	2017-12-14 15:03:29	🔗 🌐

Jak to celé funguje?



Princip fungování – záznamy entit

■ Každá entita má v DB záznam

- **IP adresa**, BGP prefix, ASN, IP block, organizace
(pouze adresy nahlášené do Wardenu jako zdroj nějaké bezp. události)
- Záznam = JSON dokument

■ Záznam obsahuje

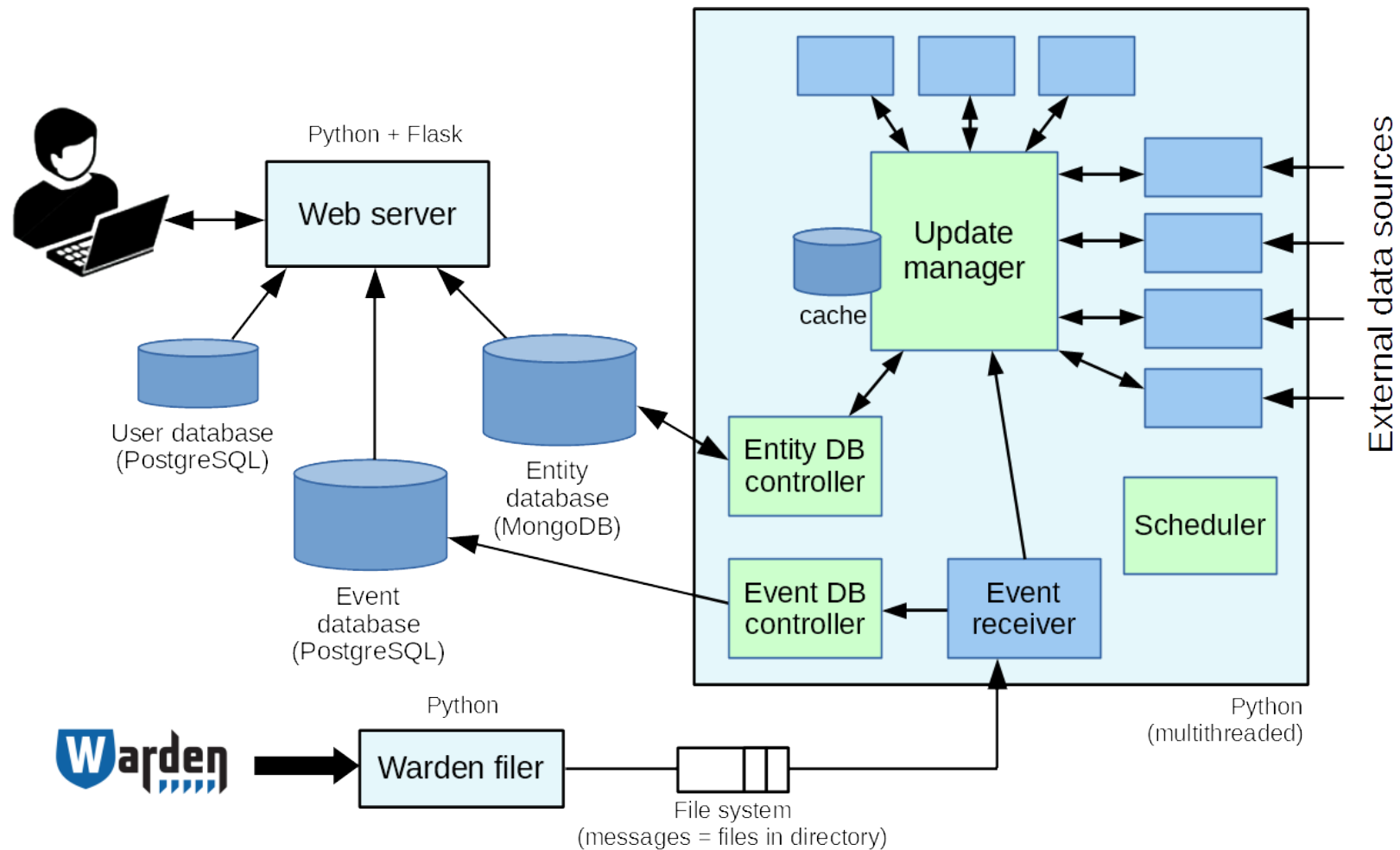
- Metainformace o nahlášených událostech
 - počet událostí pro každý den, kategorii a detektor
- Hostname, geolokace, přítomnost na blacklistech, různé tagy
- Reputační skóre
- Odkazy na BGP prefix, ASN atd., do kterých adresa patří
- ...

■ Všechny informace jsou pravidelně aktualizovány

■ Když k dané adrese nepřijde žádná událost po 14 dní, je záznam smazán

- Systém je modulární – o každou část záznamu se stará konkrétní **modul**
■ Např. modul pro geolokaci, modul pro dotazy do blacklistů, atd. (plugin?)
- Všechny změny v záznamu řídí **UpdateManager**
 - Přijímá asynchronní požadavky na změny v záznamech
 - Např. při přijetí nové události z Wardenu
 - Pravidelné aktualizace záznamů
 - Volá funkce z jednotlivých modulů
 - ... a zapisuje všechny změny do DB

Současná architektura a základní princip fungování



■ Entity key (ekey)

- Identifikátor záznamu, dvojice (entity_type, entity_id)
- Např.: ("ip", "1.2.3.4"), ("asn", 2852)

■ Atribut záznamu

- Název (klíč) položky v JSONu
- Např.: "rep", "geo.ctrý"

■ Update request

- Žádost o nějakou změnu v záznamu (či vyvolání pojmenované „události“)
- 3-tuple (operace, klíč, hodnota/parametr)
- Např.:
("set", "geo.ctrý", "CZ")
("add", "events_meta.total", 1)
("event", "!every1d", None) # pozn.: název události by měl vždy začínat "!"
- Událost (event) přímo nic nemění
- Speciální událost !NEW – generována automaticky při přidání nového záznamu

■ Handler function

- Callback funkce zaregistrovaná na změnu určitého atributu či událost
- Zpravidla definovány v modulech
- Prototyp:

```
def <name>(self, ekey, rec, updates):
```
- Vrací `None` či seznam dalších update requestů, které se mají na danou entitu aplikovat
- Např. geolokační modul má funkci `geoloc` zaregistrovanou na událost `!NEW`, funkce zjistí lokační údaje dané IP adresy a vrátí 3 update requesty:

```
[  
    ('set', 'geo.ctrý', ctry),  
    ('set', 'geo.city', city),  
    ('set', 'geo.tz', tz)  
]
```

■ Hlavní fronta update requestů

- Obsahuje záznamy ve formátu

```
(entity_key, [(op, key, value), (op, key, value), ...])
```
- Jakýkoliv modul do ní může asynchronně zapisovat

- 1) Načte požadavek z hlavní fronty update requestů
- 2) Načte záznam dané entity (dle `entity_key`)
 - Pokud neexistuje, vytvoří nový záznam a na začátek seznamu `update_requestů` přidá událost !NEW
- 3) Proveďte požadované změny v záznamu
 - Pokud jsou na změněné atributy či události zaregistrovány nějaké handler funkce, zavolá je
 - Update requesty vrácené z handler funkcí přidá do seznamu requestů ke zpracování a opakuje 3), dokud nějaké requesty zbývají
- 4) Uloží změněný záznam do DB

(plus trochu „intelligence“, aby se nevolaly handler funkce na atributy, které se ještě můžou změnit – v takovém případě se volání odloží)

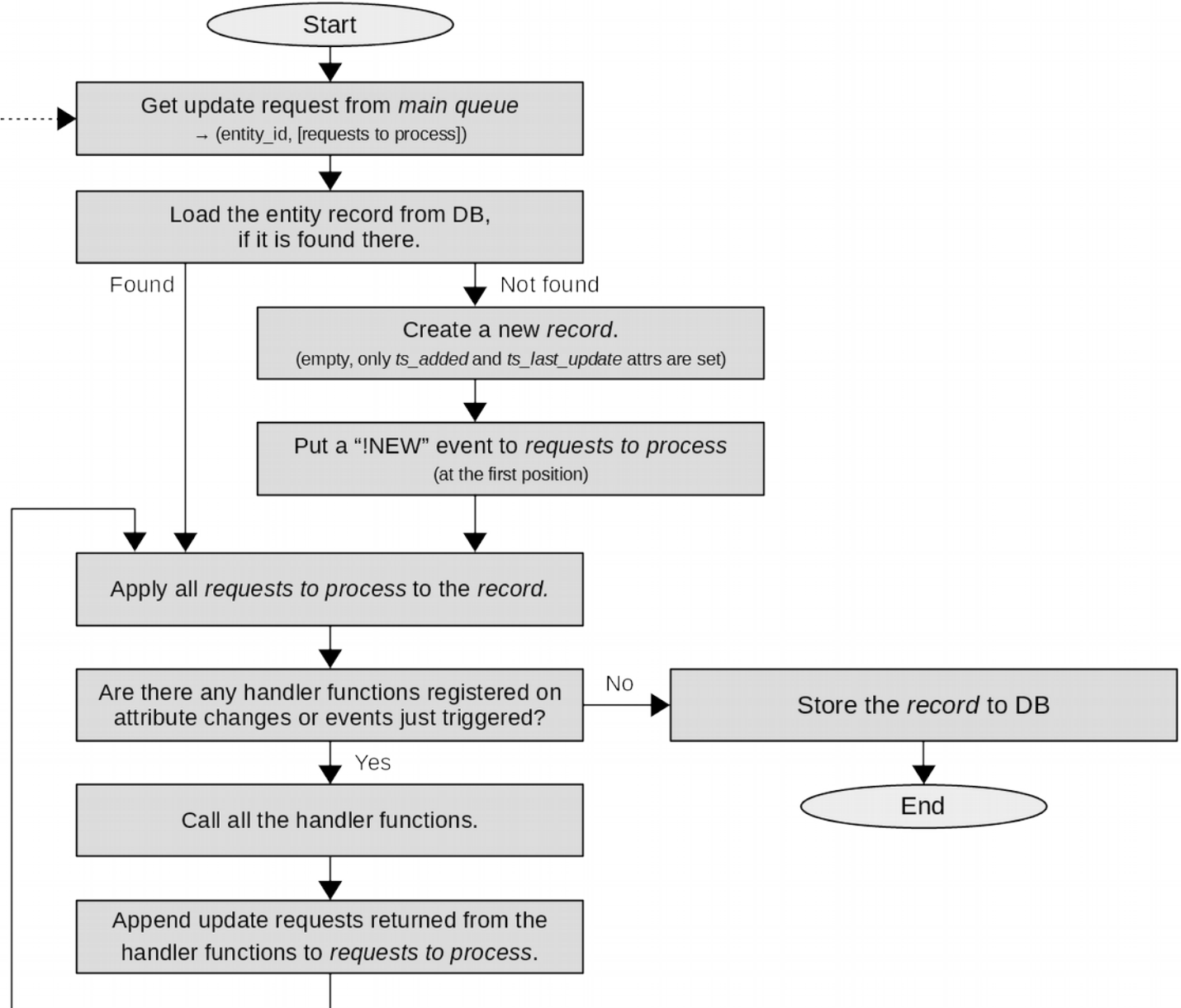
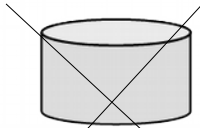
Update manager

Main processing function (runs in parallel in several threads)

Main update-request queue

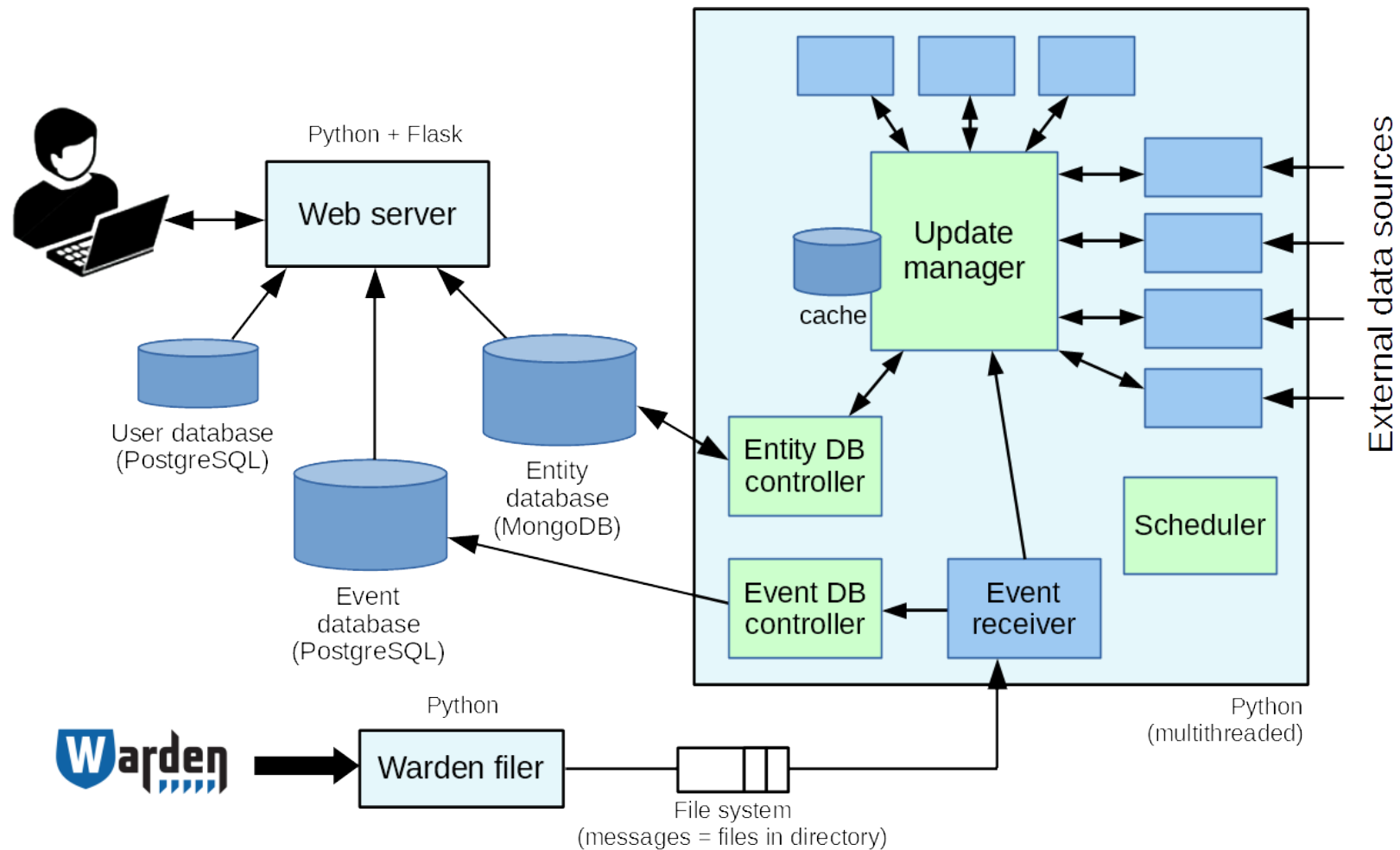


Local cache of currently processed records



- UpdateManager běží paralelně v mnoha vláknech
 - A spolu s ním tedy mohou běžet paralelně i handler funkce v modulech
 - Aby nedocházelo k tomu, že jeden záznam bude zároveň upravovat více instancí UpdateManageru, ani se nemusely záznamy zamykat, **rozděluje se práce na jednotlivá vlákna podle hashe entity_key**. Každá entita je tak vždy zpracovávána jednou konkrétní instancí UpdateManageru a tedy sekvenčně.
- Další poznámky:
 - Pokud chce handler funkce provést změny v jiné než aktuální entitě, musí vložit nový update_request do hlavní fronty.
 - Každý modul může vytvořit vlastní vlákno, které běží bez ohledu na handler funkce.
 - Použito např. pro příjem zpráv z Wardenu

Současná architektura a základní princip fungování



- Umožňuje zaregistrovat pravidelné volání jakékoliv funkce
 - (wrapper nad Python modulem **apscheduler**)
 - Takový interní „cron“
 - Použito např. modulem **Updater**, který pravidelně pro každou entitu vydává události `!every1d` a `!every1w` (jednou za den/týden). Ostatní moduly mohou zaregistrovat své handler funkce na tyto události a tak provádět pravidelné updaty (díky tomu navíc všechny updaty jedné entity probíhají současně, což je výhodné z hlediska počtu přístupů do DB).

- NERD se konfiguruje pomocí souborů ve formátu YAML
 - Strukturovaná data jako např. JSON, ale syntaxe pohodlnější pro ruční psaní + podpora komentářů
- Tři hlavní soubory:
 - nerd.yml – konfigurace potřebná backendem i forntendem
 - nerdd.yml – konfigurace pouze pro backend
 - nerdweb.yml – konfigurace pouze pro frontend
 - (mohou existovat i další soubory, např. tags.yml s definicí pravidel pro přidělování tagů, do budoucna asi i další)
- V programu dostupné v `g.config` jako třída „`hierarchical_dict`“
 - Chová se jako normální Python `dict`, ale místo `config["whois"]["asn_file"]` lze napsat `config["whois.asn_file"]`
 - Příp. u volitelných parametrů s defaultní hodnotou `config.get("whois.asn_file", "/tmp/nerd-whois-asn.csv")`

Uspořádání repozitáře



- <https://github.com/CESNET/NERD/>
- [NERDd](#) – (NERD daemon) zdrojáky backendu, tedy vše o čem jsme zatím mluvili
- [NERDweb](#) – zdrojáky webového rozhraní a API
- [common](#) – části společné pro backend a frontend
- [scripts](#) – různé pomocné skripty
- [etc](#) – konfigurační soubory
- [doc](#) – zdrojáky schémat z dokumentace
- Pozn: Dokumentace (značně nekompletní) je v github wiki:
<https://github.com/CESNET/NERD/wiki>

■ /NERDd/

- core/ – základní komponenty
- modules/ – moduly
- nerdd.py – hlavní část programu (spustitelné)
- g.py – modul s globálními proměnnými
 - Jsou zde uloženy reference na načtenou konfiguraci a instance základních komponent, např. `g.config`, `g.um`, `g.scheduler`

■ /NERDweb/

- static/ – obrázky, css, js
- templates/ – Jinja2 šablony pro generování HTML
- nerdweb.py – hlavní kód webu i API (zatím je téměř vše zde)
- wsgi.py – wrapper pro WSGI (sem míří konfigurace web serveru)
- userdb.py – vše kolem uživatelských účtů
- ctrydata.py – statická data s názvy zemí

■ Pozn.:

- Web je implementován v Python knihovně Flask.
- Bližší popis ale v této prezentaci není.

■ /common/

- config.py – třída reprezentující načtenou konfiguraci
- eventdb_psql.py – wrapper nad databází událostí (IDEA zpráv z Wardenu)
- utils.py – různé drobné funkce (aktuálně jen parser času v RFC3339)
- (eventdb.py – stará verze databáze událostí, nepoužívaná)

■ /scripts/

- Projít v repozitáři ...

■ Práce s gitem

- Drobné a řádně otestované změny je možné dávat přímo do master
- Pro větší změny vytváříme vlastní větve, které se pak do master mergují
- Pokud se nedohodneme jinak, udělejte po dokončení práce pull request, já ho zkontroluju/okomentuju a když je vše pořádku, mergnu
- Commit messages:
 - Začínají vždy „navez_modulu:“, příp. „NERDd:“, „NERDweb:“, „scripts:“ apod.
 - Následuje stručný popis změn
 - Dobré příklady:
 - „Updater: fixed bugs preventing ASNs from updating“
 - „API: added support for bulk queries of IP address reputation (#17)“
 - Špatný příklad:
 - „Fixed exception when division by zero“

Aktuální sada modulů

Jaká data se ukládají



■ EventReceiver

- Přijímá události s Wardenu
- Ukládá kopie do EventDB (PostgreSQL)
- Vydává update requesty pro úpravy položek v events a event_meta
- Běží ve vlastním vlákně, nemá žádnou handler funkci

■ Updater

- Pravidelně vydává události `!every1d` a `!every1w` pro každou entitu
- Rozloženo v čase
- Do záznamů jsou přidány položky `_nru1d`, `_nru1w` (*next regular update*)
 - obsahují čas dalšího plánovaného updatu
 - Updater každých pár sekund vyčte z DB všechny entity s `_nru*` > aktuální čas a pro každou vydá příslušnou událost

■ Cleaner

- Pročišťuje záznamy od starých informací
 - Meta informace o událostech starších než 90 dní
 - Podobně pro blacklisty
- Nechává smazat celé staré záznamy

■ Projít další moduly přímo v repozitáři ...

■ Projít datový model ...

- <https://github.com/CESNET/NERD/wiki/Attributes>

Vytvoření vlastního modulu



1) Nový soubor v /NERDd/modules/

- Obsahuje jednu třídu pojmenovanou podle funkce modulu (dědí z `BaseModule`)
- V `__init__` může zaregistrovat jednu nebo víc handler funkcí
`g.um.register_handler(self.func_name, ent_type, triggers, changes)`
(viz `register_handler` v `update_manager.py`)
- Může definovat funkce `start()` a `stop()`, ve kterých může např. vytvořit/zrušit nové vlákno

2) Přidání modulu v `nerdd.py`

- Import modulu
- Přidání instance třídy do `module_list`

■ NERD používá standardní Python knihovnu **logging**

- Každý modul si v `__init__` vytvoří vlastní logger

```
self.log = logging.getLogger("ModuleName")
```

- Logy pak lze vypisovat takto:

```
self.log.info("This is log message number {}".format(i))
```

- Výsledek:

```
2018-03-13T13:15:54,UMWorker-13,ModuleName,[INFO] This is log message number 5.
```

- Funkce loggeru:

- `.debug()`
- `.info()`
- `.warning()`
- `.error()`

- Defaultně se loguje od `INFO` výš

- Override pro vlastní modul: `self.log.setLevel("DEBUG")`

■ Další poznámky:

- Docstring souboru i třídy by měl popisovat, co modul dělá.
- Pokud modul potřebuje konfiguraci, stačí do `nerd.yml` nebo `nerdd.yml` přidat další záznam
 - V modulu bude dostupný v `g.config["asdf.qwer.xyz"]`
- Handler funkce v modulu by měly pracovat jen s konkrétním záznamem
 - Můžou běžet paralelně ve více vláknech (do budoucna procesech) - pozor na jakákoliv sdílená data
 - Záznam entity se nikdy nemění přímo – pouze prostřednictvím vrácení seznamu update requestů
 - Pokud je třeba změnit (vytvořit) jinou entitu:

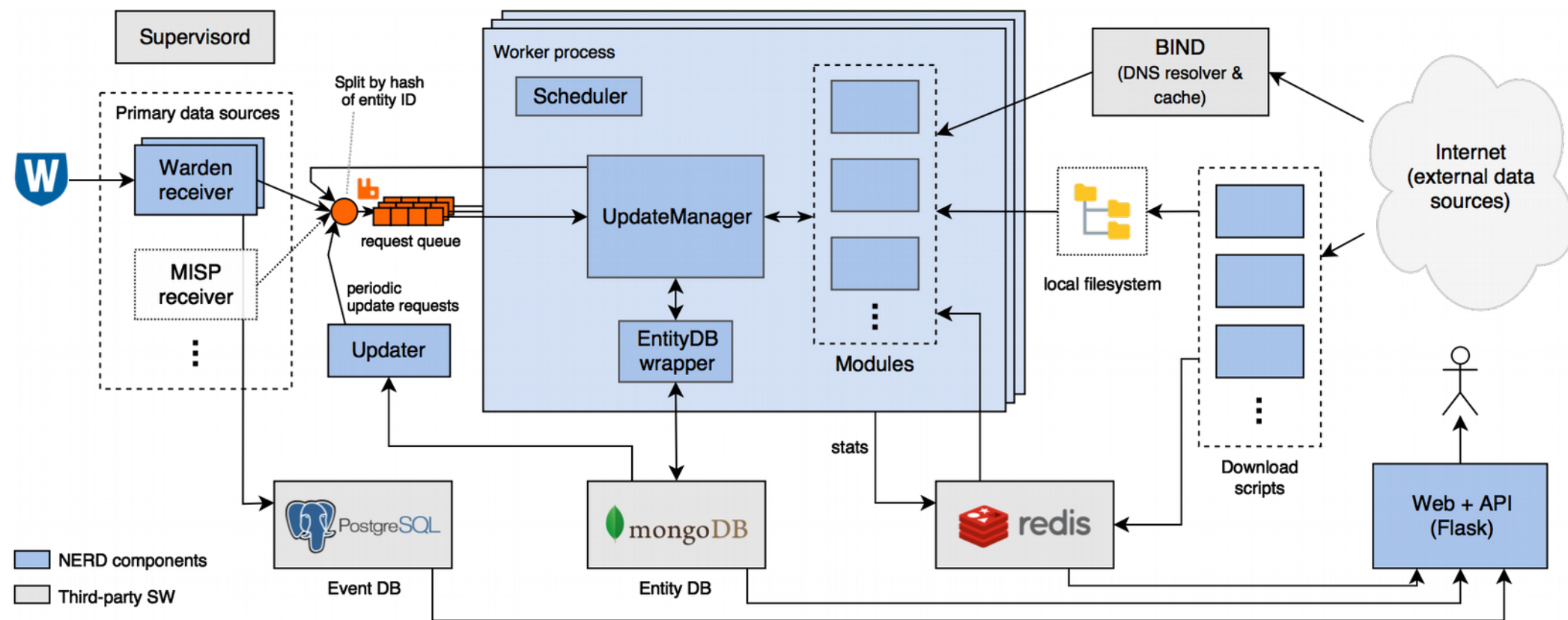
```
g.um.update(entity_key, list_of_update_requests)
```
- Není problém přidat závislost, pokud je to balíček v PyPI
 - Přidat do `/NERDd/requirements.txt`

Plány do blízké i vzdálenější budoucnosti



- Nedávno začal přechod na novou, lépe škálovatelnou architekturu
 - Ve staré běží celý backend v jednom procesu; sice v mnoha vláknech, ale v Pythonu to neznamena opravdovou paralelizaci
 - V nové jde spíš o množství spolupracujících procesů
 - Snadno distribuovatelné klidně i na více serverů

Nová architektura



Popis a seznam potřebných úprav: <https://github.com/CESNET/NERD/wiki/Parallelization>

■ Hlavní změny

- Primární zdroje dat → speciální případ, ne jen běžný modul
- Moduly běží paralelně ve více procesech → problém mít společný stav
 - (např. načtená data ze souboru)
- Pro sdílená data Redis
- Pro komunikaci mezi komponentami RabbitMQ

■ Lenka Stejskalová

- Databáze „skupin IP adres“ (botnetů), shlukování na základě podobnosti hlášení
- Zatím samostatný systém, ale časem by mělo být více či méně integrované s NERD

■ Maxmilián Tomáš

- PassiveDNS

■ Ladislav Macoun

- Analýza historického vývoje bezp. událostí (neznám podrobnosti)

■ Tomáš Jánský

- Integrace s DDoS čističkou (využití reputačního skóre z NERD pro mitigaci DDoS útoků)

■ Tomáš Kukaň

- Různé drobnosti, co jsou zrovna potřeba

■ Další ?

- Zapojení dalších primárních zdrojů
 - MISP, OV ATX, ...
- Ukládání všech IP z blacklistů
- Integrace s PassiveDNS, Negistry, ActivityMonitor
- On-demand zjišťování dat o neznámých adresách na dotaz z frontendu
- Hromadné dotazy z frontendu
- Timeline IP adresy ve frontendu
- Celkový redesign frontendu
- Možnost vyhledávání zpráv z Wardenu
 - Přístup přímo do nové DB Mentatu

■ Veřejná verze NERD

- Zobrazující jen data z veřejných zdrojů, data z Wardenu jen v hodně agregované podobě

■ Plugin pro prohlížeče

- Přidá ke každé IP adrese na stránce pop-up okénko s informacemi z NERDu
- Příp. nativní aplikace pro různé OS, sledující schránku, a pokud se tam objeví text s IP adresou, vyskočí notifikace s informacemi z NERD

■ Multiple-instance / hierarchický NERD (?)

- Jeden master NERD s informacemi pro všechny + více podřízených instancí, kam si každý bude moct přidat další zdroje dat, které ale nechce sdílet s ostatními
- Zatím netuším, jak to udělat

■ Automatický reporting (?)

- Posílání upozornění správcům, že jejich adresa má špatnou reputaci
- Principiálně není problém a lidi to chtějí, ale nechce se mi :)



DĚKUJI ZA POZORNOST

MÁTE NĚJAKÉ DOTAZY?