UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

VIZUALIZÁCIA ČASOVO-ORIENTOVANÝCH DÁT V HONEYNETE

2015 **Lenka KLEINOVÁ**

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

VIZUALIZÁCIA ČASOVO-ORIENTOVANÝCH DÁT V HONEYNETE

BAKALÁRSKA PRÁCA

Študijný program:	Informatika
Pracovisko (katedra/ústav):	Ústav informatiky
Vedúci bakalárskej práce:	RNDr. JUDr. Pavol Sokol

Košice 2015 **Lenka KLEINOVÁ**

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Lenka Kleinová
Študijný program: Informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: Bakalárska práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Vizualizácia časovo-orientovaných dát v honeynete

Cieľ:
1. Analyzovať možnosti vizualizácie časovo-orientovaých dát pri honeypotoch a honeynetoch
2. Porovnať aktuálne prístupy k vizualizácii časovo-orientovaých dát v honeypotoch a honeynetoch
3. Navrhnúť a implementovať vizualizáciu priebehu útoku na honeynet

Literatúra:
[1] Kirk, A.: Data Visualization: A Successful Design Process. Packt, 2012.
[2] Wolfgang,A. et al.: Visualization of time-oriented data. Springer, 2011.
[3] Joshi, R. C., Sardana,A.: Honeypots: A New Paradigm to Information Security. Science Publishers, 2011.
[4] Provos, N., Thorsten H.: Virtual honeypots: from botnet tracking to intrusion detection. Addison-Wesley Professional, 2007.

Kľúčové slová: honeypot, honeynet, časovo-orientované dáta, čas, vizualizácia

Vedúci: RNDr. JUDr. Pavol Sokol
Ústav: ÚINF - Ústav informatiky
Riaditeľ ústavu: prof. RNDr. Viliam Geffert, DrSc.

Spôsob sprístupnenia elektronickej verzie práce:
dočasne neprístupná, po uplynutí bez obmedzenia

Dátum zadania: 17.05.2014

Dátum schválenia: 07.05.2015

prof. RNDr. Viliam Geffert, DrSc.
riaditeľ ústavu

Vyhlásenie

Vyhlasujem, že som túto bakalársku prácu vypracovala samostatne na základe vedomostí získaných štúdiom a s pomocou uvedenej literatúry.

_____

Lenka Kleinová

Poďakovanie

Týmto by som sa rada poďakovala vedúcemu svojej práce RNDr. JUDr. Pavlovi Sokolovi za pomoc, trpezlivosť a cenné rady pri vypracovaní tejto práce.

Abstrakt v štátnom jazyku

Práca sa venuje problematike honeypotov a honeynetov. Základnou úlohou týchto pascí na útočníkov je získavanie údajov a následná analýza týchto údajov. V práci sa venujeme práve analytickej časti honeypotov a honeynetov a rozoberáme možnosti, ako vhodne a pre administrátora čo najprehľadnejšie zobrazovať údaje získané pomocou jednotlivých honeypotov v honeynete. V práci sa zameriavame na vizualizáciu časovo-orientovaných údajov, čo sú údaje, ktorých jeden rozmer definičného oboru je asociovaný s časom. Keďže každý údaj zbieraný honeypotmi je asociovaný s časovou pečiatkou, považujeme údaje získavané honeypotmi za časovo-orientované údaje. V rámci práce rozoberáme niekoľko vizualičných metód, ktoré sú vzhľadom na údaje zbierané honeypotmi, najviac použiteľné. Tieto metódy následne porovnávame s rôznymi vizualizačnými prostrediami (frameworkami) a vyberáme tie, ktoré sú pre zvolené vizualizačné metódy najvhodnejšie. Výsledkom práce je návrh a implementácia webovej aplikácie, ktorá zobrazí pomocou zvolenej vizualizačnej metódy informácie o útoku, najmä o jeho priebehu. Ďalším výsledkom práce je analýza reálnych údajov zozbieraných honeypotmi (najmä HoneyD) počas roka 2014. Implementácie jednotlivých vizualizačných metód je možné priamo nasadiť v systémoch na analýzu údajov z honeypotov alebo honeynetov. **Kľúčové slová:** honeypot, honeynet, časovo-orientované údaje, čas, vizualizácia.

Abstract

The thesis is dedicated to the issue of honeynets and honeypots. The main goal of these traps is to lure the attackers, collect and analyze the data about their activity. In this thesis we focus on the analytic aspects of honeypots and honeynets and investigate the possibilities how to display the collected data to an administrator in efficient and readable way. We focus on the visualization of time-oriented data which is the data with at least one dimension associated with the dimension of time. Since every entry collected with honeypots is associated with timestamp we call this data time-oriented. We analyze several visualization methods which are most suitable for data collected with honeypots, compare them with different visualization frameworks and choose those ones which are most suitable for the chosen visualization methods. The result of the thesis is implementation of the web application which displays the data from honeynet using the chosen visualization method. Another result is the analysis of the real data collected with honeypots during the year 2014 (especially HoneyD). It is possible to use all of these implementations in real systems for analysis of data from honeypots or honeynets. **Key words:** honeypot, honeynet, time-oriented data, time, visualization.

Obsah

1 Úvod

Priestor a čas sú dimenzie, spojením ktorých vzniká štvordimenzionálny priestor alebo inak povedané svet, v ktorom žijeme. Všetky merateľné údaje majú väčšinou význam len vtedy, ak sú uvedené v kontexte priestoru a času. V priestore sa vieme pohybovať- vrátiť sa späť či prejsť pár krokov vpred. Avšak nemôžeme ísť späť v čase, ani sa pozrieť do budúcnosti. Navyše nemáme zmysly, ktorými by sme priamo dokázali vnímať čas. Preto ho potrebujeme vizualizovať a spraviť tak neviditeľné viditeľným.

Sieťová bezpečnosť je v súčasnom svete informačných technológií jedným z kľúčových pojmov. Ak je v našom záujme dbať o bezpečnosť siete, serverov či počítačových staníc, je potrebné sledovať aktuálne trendy v oblasti hrozieb informačných technológií (IT). Honeypot je prostriedok, ktorý nám umožní lepšie porozumieť a analyzovať „druhú“ stranu, teda útočníka, jeho postupy či nástroje, ktoré využíva. Zmysel honeypotu spočíva v napadnutí, teda vo využití nejakej škodlivej činnosti. Každé pripojenie na takýto honeypot je považované za útok. Preto je potrebné zabezpečiť to, aby útočník v žiadnom prípade netušil, že útočí na honeypot. Sieť honeypotov, teda honeynet, slúži na analýzu činnosti útočníka v oveľa vyššej miere ako by to dokázal samotný honeypot. Takto získané údaje môže následne prevádzkovateľ siete spracovávať a vyhodnocovať.

Na to, aby sme si o získaných údajoch vedeli vytvoriť hypotézu v čo najkratšom čase, slúži vizualizácia. Podľa definície časovo-orientovaných údajov platí, že ak aspoň jeden rozmer definičného oboru je asociovaný s časom, tak tieto údaje nazývame časovo-orientované údaje. Väčšina údajov získaných pomocou honeypotov je spätá s časovou pečiatkou, ide teda o časovo-orientované údaje. Príkladom sú informácie o tom, aké bolo v danom čase vyťaženie procesora, využitie pamäte, koľko bolo otvorených portov či napríklad koľko pokusov o útok bolo v danom čase zaznamenaných na konkrétny honeypot. Všetky tieto údaje sa ukladajú do centrálnej databázy a sú určené na ďalšie spracovanie. Vďaka tomu tak získavame informácie o útočníkovi, čo zahŕňa napríklad pokusy o prístup do systému, všetky stlačené klávesy klávesnice, súbory, ku ktorým sa pristúpilo a ktoré boli modifikované a vykonané procesy.

Práca je rozdelená na štyri základné časti. V prvej z nich sa venujeme honeypotom a honeynetom, ich klasifikácii, výhodám a nevýhodám. Druhá časť práce

je venovaná vizualizačným metódam časovo-orientovaných údajov, t. j. aké sú ich všeobecné vlastnosti, kedy je možné využiť jednotlivé vizualizačné techniky a ako ich možno využiť na zobrazenie údajov z honeynetu. Tretia časť analyzuje a porovnáva už existujúce riešenia v oblasti vizualizácie časovo-orientovaných údajov. Posledná- štvrtá časť práce- sa venuje vlastnej implementácii vybraných vizualizačných metód na konkrétne údaje.

2 Honeypoty a honeynety

2.1 Definícia honeypotu

Honeypot môže byť definovaný viacerými spôsobmi. Jedna z definícií hovorí, že honeypot je program, ktorý na prvý pohľad vyzerá ako atraktívna služba, či skupina služieb, celý operačný systém alebo dokonca celá počítačová sieť, no v skutočnosti ide len o systém, ktorého úlohou je nalákať útočníka a monitorovať jeho činnosť[12]. Honeypot teda monitoruje a zaznamenáva každý krok, ktorý útočník urobí, čo zahŕňa pokusy o prístup do systému, všetky stlačené klávesy klávesnice, súbory, ku ktorým sa pristúpilo a ktoré boli modifikované a vykonané procesy. Lance Spitzner, zakladateľ Honeynet Project-u, definuje honeypot nasledovne: „Honeypot je informačný systém, ktorého hodnota spočíva v jeho neautorizovanom alebo nedovolenom využití.“ [10]

Na honeypote je spustený emulovaný operačný systém a služby, ktoré sa správajú ako „pasca“ a ich úlohou je nalákať útočníka, aby zneužil systém na nejakú nekalú činnosť, pričom v skutočnosti honeypot len zaznamenáva všetky prostriedky, ktoré útočník na túto činnosť využíva[12].

2.1.1 Generický model honeypotu

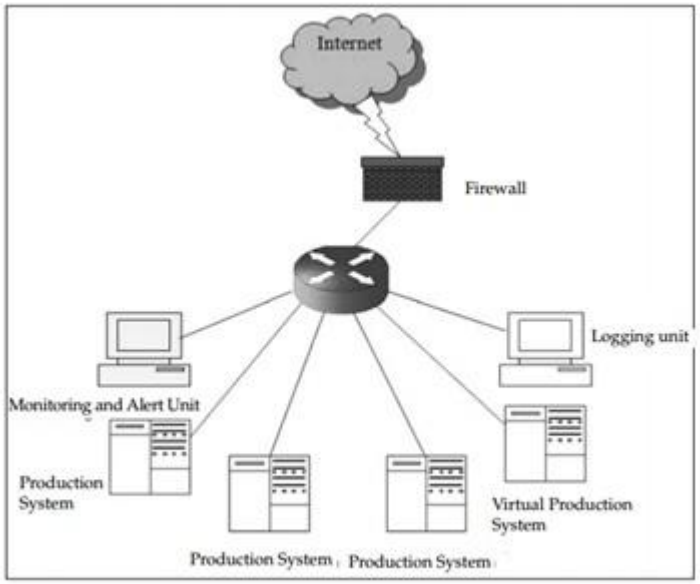
Honeypot je umiestnený do siete s jediným účelom, a to byť napadnutý. Je navrhnutý s úmyselnými nechránenými miestami, ktoré sú odhalené vo verejnej sieti. Honeypot nemá žiadnu produkčnú hodnotu a každý prístup k nemu je považovaný za nelegálny.

Honeypot obsahuje:[12]

- **Produkčný systém honeypotu-** nejde o skutočný produkčný systém, len o akúsi „korisť“ pre útočníka. Poskytuje honey (med), teda súbory a falošné systémové prostriedky a na každú útočníkovu aktivitu je nastavená automatická odpoveď, aby honeypot vyzeral ako skutočný produkčný systém.
- **Firewall-** poskytuje záznamy o tom, ako sa útočník pokúsil dostať do systému honeypotu. Zaznamenáva všetky pakety idúce do systému, pretože každý prístup do honeypotu má nejaký nelegálny dôvod.
- **Monitorovacia jednotka-** ide o jednotku vyhodnocujúcu ohrozenie, ktorá monitoruje aktivity v sieti a/alebo v systéme a zisťuje tak škodlivú činnosť.

O každej takejto činnosti informuje riadiacu stanicu. Prehodnocuje poradie, časové pečiatky a typ paketov, ktoré útočník použil na získanie prístupu do honeypotu a stlačené klávesy či zmenené súbory pomáhajú identifikovať prostriedky, metodiku a zámery útočníka. Ako monitorovacia jednotka môže slúžiť systém na detekciu útoku (intrusion detection system-IDS).

- **Výstražná jednotka**- Honeypot by mal byť schopný generovať upozornenie, výstrahu cez email a poslať administrátorovi upozornenie o toku údajov z alebo do honeypotu. Tak môže administrátor skúmať útočníkovu aktivitu ešte počas doby, keď prebieha.
- **Zaznamenávacia jednotka**- poskytuje efektívne uchovanie všetkých systémových záznamov, záznamov firewallu a záznamov o toku údajov medzi firewallom a honeypotom.



Obrázok 1 Generický model honeypotu [12]

2.2 Výhody a nevýhody honeypotov

V oblasti bezpečnosti majú honeypoty od svojho vzniku postavenie úspešného nástroja a technológie využívanej v spojení s Intrusion Detection System (IDS) a firewallmi. Napriek mnohým výhodám, honeypot ako mechanizmus včasnej detekcie podozrivej činnosti v sieti so sebou prináša aj riziká, a to najmä riziko útoku v sieti, do ktorej je honeypot nasadený.

Medzi dôležité výhody honeypotov patrí:[12]

1) Údaje zbierajú len vtedy, keď niekto alebo niečo s nimi interaguje. To robí údaje, ktoré honeypot zbiera jednoduchšie na spracovanie a analýzu.
2) Honeypoty výrazne redukujú počet falošných výstrah. Akákoľvek aktivita s honeypotom je z jeho definície neautorizovaná, čo robí detekciu útokov omnoho efektívnejšiu. Honeypoty nemajú žiadnu produkčnú hodnotu a nikdy by nemali byť použité nikým iným ako administrátorom. Každý tok údajov do honeypotu okrem očakávaného administratívneho toku je pravdepodobne škodlivý. Každý tok údajov z honeypotu je škodlivý. Preto sa v údajoch nebude nachádzať žiaden, resp. veľmi malý šum a preto by všetko, čo honeypot zachytí, malo byť analyzované. Tento nízky počet falošných výstrah je oproti IDS alebo firewallu veľkou výhodou honeypotov. Vďaka tomu je možná rýchla detekcia hrozieb a generovanie upozornenia.
3) Keďže každé pripojenie na honeypot je považované za hrozbu, tak dosiaľ nepoznané útoky sú odhalené tak rýchlo ako aj tie, ktoré sú už známe. Hovoríme o takzvaných „zero-day“ hrozbách, ktoré sú odhaľované vďaka honeypotom. Honeypoty dokážu zachytiť všetko týkajúce sa útočníka, čo zahŕňa sieťové pakety, uploadovaný malware, všetku komunikáciu cez chat a všetky príkazy. Tak vie administrátor zistiť, čo útočník robí a ako to robí.
4) Ďalšou výhodou je jednoduchosť. Neexistujú tu žiadne špeciálne algoritmy, ktoré by bolo treba vyvíjať, netreba udržiavať žiadne stavové tabuľky, ani obnovovať podpisy kvôli šifrovaniu. Vyžadujú minimálne prostriedky, dokonca i vo veľmi veľkých počítačových sieťach.
5) Na rozdiel od mnohých iných bezpečnostných technológií, ako je napríklad IDS, honeypoty dokážu fungovať aj v šifrovanom alebo v IPv6 prostredí. Preto nezáleží na tom, čím útočník zaútočí, honeypot to zachytí.

Rôzne typy honeypotov so sebou prinášajú rôzne riziká. Medzi ne patria aj nasledovné: [12]

1) Pokiaľ na honeypoty nik neútočí, sú zbytočné.
2) Honeypoty vidia len to, čo je namierené priamo na ne. To znamená, že ak sa útočník dostane do siete a zaútočí na systémy v nej okrem daného honeypotu, tento honeypot nebude vedieť nič o útočníkovej aktivite pokiaľ nie je útok namierený práve naň. Okrem toho, ak útočník identifikuje honeypot, t.j. zistí, že

nejde o skutočný produkčný systém, môže sa honeypotu úspešne vyhýbať a napadnúť organizáciu bez toho, aby o tom honeypot vedel.

3) Útočník môže využiť honeypot aj vo svoj prospech. Ak úspešne vnikne do tohto systému, môže ho ďalej využívať na ďalšie útoky (**zneužitie honeypotu**).
4) Ďalším rizikom honeypotov je takzvaný **fingerprinting**. Ide o to, že útočník vie identifikovať, že ide o honeypot, pretože systém, o ktorom si útočník myslí, že je produkčný, má určité očakávané charakteristiky alebo správanie, ktoré honeypot v dôsledku zlého nastavenia nemá.
5) Fingerprinting je veľkou hrozbou najmä pre výskumné honeypoty. Po tom, čo útočník odhalí, že ide o honeypot, môže zásobovať daný honeypot zlými, zavádzajúcimi informáciami, čím privedie výskumný bezpečnostný tím k **nesprávnym záverom**.

Kvôli týmto nevýhodám honeypoty nemôžu úplne nahradiť iné bezpečnostné systémy. Napriek tomu však týmto systémom pridávajú hodnotu a robia ich účinnejšími.

2.3 Klasifikácia honeypotov

V predchádzajúcej kapitole sme zadefinovali pojem honeypot. V súčasnej dobe existuje niekoľko typov honeypotov, ktoré môžeme klasifikovať podľa niekoľkých kritérií. V nesledujúcich podkapitolách sa zameriavame na klasifikáciu podľa:

- použitia,
- úrovne interakcie,
- hardvérového nasadenia a
- úlohy honeypotu.

2.3.1 Klasifikácia honeypotov podľa použitia

Produkčné honeypoty

Tieto honeypoty slúžia na ochranu organizácií. Znižujú riziko napadnutia tým, že odhalia zraniteľné miesta systému a každý pokus o útok okamžite hlásia administrátorovi. Takýmto spôsobom odlákajú útočníka od skutočného cieľa, ktorého napadnutie by znamenalo veľký problém. Vďaka honeypotu organizácia, ktorú chceme chrániť, získa užitočné informácie, ktoré tak prispejú k vylepšeniu pravidiel firewallov

alebo IDS systémov. Takto je organizácia chránená ešte pred tým než bezpečnostné inštitúcie vydajú svoje aktualizácie pre rôzne bezpečnostné prvky[12].

Výskumné honeypoty

Výskumné honeypoty sa využívajú na skúmanie hrozieb, ktorým musia organizácie čeliť, a pomocou nich sa tak zisťuje, ako lepšie chrániť pred týmito hrozbami. Tieto honeypoty sú väčšinou prevádzkované univerzitami, vládou, armádou či rôznymi bezpečnostnými výskumnými organizáciami. Výskumné honeypoty sú výborným nástrojom pre zachytávanie a analýzu automatizovaných útokov ako sú napríklad červy. Napríklad The HoneyNet Project je nezisková bezpečnostná výskumná organizácia, ktorá využíva honeypoty na zbieranie informácií o elektronických hrozbách[12].

2.3.2 Klasifikácia honeypotov podľa úrovne interakcie

Nízko-interaktívne honeypoty

Sú charakteristické minimálnou interakciou s útočníkom a emulovaním falošných služieb ako napríklad ftp alebo http. Nebeží na nich žiaden skutočný operačný systém ani služby, ide len o emulácie bežiace nad vrstvou operačného systému. Sú jednoduché na nasadenie aj údržbu, ale dokážu zaznamenávať len obmedzené množstvo informácií o útočníkovej aktivite. Keďže služby bežia nad vrstvou operačného systému, tak operačný systém je chránený pred možným prebratím kontroly útočníkom. Maximálne poškodenie, ktoré môže útočník spôsobiť sa týka práve týchto emulácií honeypotu. Nízko-interaktívne honeypoty sú užitočné najmä pri identifikovaní útočníkovej IP adresy[12]. Patrí sem napríklad **Honeyd** [32] alebo **Specter**[33].

Stredne-interaktívne honeypoty

Kombinujú výhody nízko-interaktívnych honeypotov a vysoko-interaktívnych honeypotov. Tieto honeypoty nemajú ani skutočný operačný systém, ani neimplementujú všetky detaily aplikačného protokolu, ale majú vrstvu virtualizácie. Poskytujú odpovede, ktoré útočník očakáva. Tieto odpovede sa využívajú na nalákanie útočníka, aby zaťažil systém a aby sa jeho činnosť dala ďalej analyzovať. Stredne-interaktívne honeypoty sú zložité, náročné na nasadenie a vyžadujú množstvo poznatkov o protokoloch, aplikačných službách a bezpečnosti. Okrem toho je pri nich viac rizík ako pri nízko-interaktívnych honeypotoch [12]. Príkladom je honeypot **Kippo** [34].

Vysoko-interaktívne honeypoty

Tieto honeypoty poskytujú skutočný operačný systém, ktorý veľmi dobre láka útočníka, no na druhej strane je systém vystavený veľkému riziku. Vysoko-interaktívne honeypoty sa využívajú najmä pre výskumné účely, keďže poskytujú úplný prístup čím vzrastá atraktívnosť honeypotu a tým aj možnosť získať čo najviac informácií o útoku. Sú veľmi náročné na nasadenie, pretože ich fungovanie vyžaduje množstvo nástrojov. Avšak po úspešnom nasadení je takýto honeypot užitočný pri odhaľovaní nových červov, vírusov a napadnuteľných miest. Príkladom je **Sebek** [35]**, Quebek** a **HonSSH**.

2.3.3 Klasifikácia honeypotov podľa hardvérového nasadenia

Fyzické honeypoty

Ide o samostatný stroj, na ktorom beží skutočný operačný systém a skutočné služby, honeypot je pripojený do siete a je prístupný cez samostatnú IP adresu. Fyzické honeypoty sú spojené s konceptom vysoko-interaktívnych honeypotov. Sú menej praktické kvôli náročnej údržbe [12].

Virtuálne honeypoty

Takýto honeypot je väčšinou implementovaný ako samostatný fyzický stroj, na ktorom beží niekoľko virtuálnych honeypotov. Sú veľmi efektívne pri monitorovaní veľkého počtu IP adries a emulovaní veľkého počtu IP adries naraz[12]. Príkladom virtuálneho honeypotu je AGROS.

2.3.4 Klasifikácia podľa úlohy honeypotu

Serverové honeypoty

V tomto prípade honeypot pasívne čaká a ponúka svoje zraniteľné služby ako server. Sú užitočné pri detegovaní a analýze nových hrozieb či pri zbieraní malwaru[10]. Príkladom sú nízko-úrovňové honeypoty a Honeyd[32].

Klientské honeypoty

Ide o aktívne honeypoty, ktoré aktívne prehľadávajú Internet a interagujú so servermi. Klientské útoky sú útoky, ktorých cieľom sú klientské aplikácie ako napríklad webové servery, kde klient interaguje so škodlivými servermi. Cieľom týchto honeypotov je nájsť tieto škodlivé servery[10]. Príkladom klientského honeypotu je Strider Honey Monkey.

2.4 Definícia honeynetu

Honeynety sú pomerne nový koncept vo svete honeypotov. Honeypoty sú známe od roku 1990, no vývoj honeynetu sa začal až v roku 1999, keď Lance Spitzner navrhol jeho koncept. Dva alebo viac honeypotov v sieti vytvárajú honeynet. Honeynet sa využíva na monitorovanie väčšej a rozmanitejšej siete, v ktorej by jediný honeypot nebol postačujúci[12].

Honeynet je vysoko-interaktívny honeypot. Koncept honeynetu je relatívne jednoduchý. Vyžaduje vybudovanie siete štandardných produkčných systémov. Tieto systémy sú umiestnené za nejaké zariadenie kontrolujúce prístup, ako napríklad firewall, a sledujú sieť. Útočníci môžu skúšať a útočiť na ktorýkoľvek systém v rámci honeynetu, ktorý im poskytuje plný operačný systém a aplikácie, s ktorými môžu interagovať. Žiadna zo služieb nie je emulovaná, ale systémy v rámci honeynetu sú skutočné produkčné systémy[12].

Honeynety sú jednoduché mechanizmy, ktoré fungujú na rovnakom princípe ako honeypoty. V prvom rade je potrebné vytvoriť zdroj, ktorý disponuje malým alebo žiadnym produkčným tokom údajov. Čokoľvek zaslané do honeynetu je podozrivé. Môže ísť o nejaký snímač či dokonca útok. Čokoľvek odoslané z honeynetu znamená, že honeynet je ohrozený, teda že útočník alebo nejaký nástroj začal svoju aktivitu. Aj keď koncept honeynetu je veľmi podobný honeypotu, rozdiel je v tom, že honeynet nie je samostatný systém, ale fyzická sieť viacerých systémov. Je to architektúra, ktorá vytvára vo vysokej miere kontrolovanú sieť, v rámci ktorej je možné umiestniť akýkoľvek systém či aplikáciu. Všetka aktivita zachytená v tomto kontrolovanom prostredí hovorí o tom, aké nástroje a taktiku útočníci využívajú a čo ich k útokom vedie. Dokážu získať detailné informácie o útočníkovi, ako napríklad stlačené klávesy pri útočení na systém, komunikáciu s inými útočníkmi alebo nástroje, ktoré využívajú na sledovanie systému. Vďaka honeynetu je možné vidieť krok za krokom, čo útočník vykonáva a z akého dôvodu [18].

2.5 Časti honeynetu

Architektúra honeynetu je definovaná štyrmi základnými prvkami:

- **kontrola údajov** (data control),
- **zachytávanie údajov** (data capture),
- **zber údajov** (data collection) a

- **analýza údajov** (data analysis).

Prvé dve z nich sú najdôležitejšie a týkajú sa každého nasadenia honeynetu. Tretí prvok, teda zber údajov sa týka len organizácií, ktoré majú nasadené viaceré honeynety v distribuovanom prostredí. Niektorí autori uvádzajú, že analýza údajov je nutná súčasť honeynetu. V nasledujúcich podkapitolách sa budeme podrobnejšie venovať jednotlivým prvkom.

2.5.1 Kontrola údajov

Kontrola údajov je časť honeynetu, ktorá zmenšuje akékoľvek riziko. Kontroluje útočníkovu aktivitu tým, že limituje to, čo sa môže stať v rámci prichádzajúceho či odchádzajúceho toku údajov. Rizikom je to, že ak raz útočník ohrozí systém v rámci honeynetu, tak môže využiť tento systém pre útok na iné systémy, ktoré už však nie sú súčasťou honeynetu, ako sú napríklad organizácie na internete. Útočník musí byť kontrolovaný, aby k takýmto útokom nemohlo dôjsť. K tomuto zabráneniu musí dôjsť čo najskôr, aby neboli spôsobené žiadne škody, a preto je najlepšie, aby bola odpoveď systému automatizovaná. Ďalšou výzvou je zabezpečiť to, aby si útočník neuvedomil, že jeho aktivita je kontrolovaná [10].

Na zabezpečenie kontroly údajov je potrebné splniť nasledujúcich 8 požiadaviek. Len tak je možné čo najviac znížiť možné riziká[12].

- Zabezpečenie kontroly údajov cez automatizované odpovede alebo manuálny zásah.
- Ochrana pred zlyhaním aspoň dvoma vrstvami kontroly údajov.
- Schopnosť uchovať stav všetkých prichádzajúcich aj odchádzajúcich spojení.
- Schopnosť kontrolovať všetku neautorizovanú aktivitu.
- Kontrola údajov musí byť kedykoľvek konfigurovateľná administrátorom.
- Kontrolné spojenia musia byť pre útočníka veľmi ťažko detekovateľné.
- Použitie aspoň dvoch metód upozorňovania na podozrivú aktivitu.
- Vzdialená administrácia kontroly údajov.

2.5.2 Zachytávanie údajov

Tak ako aj pri kontrole údajov, aj tu je potrebné, aby si útočník neuvedomoval, že jeho aktivita je nejakým spôsobom monitorovaná. Pre efektívne zachytávanie údajov je potrebné dodržať nasledujúce požiadavky:[12]

- Žiadne zachytené údaje nemôžu byť uložené lokálne v honeypote, keďže takto uložené údaje sú považované za nespoľahlivé a je tu možnosť, že sa k nim útočník môže dostať a modifikovať ich. Ide o akékoľvek údaje súvisiace s aktivitou v rámci prostredia honeynetu.
- Nasledujúce aktivity musia byť zachytené a archivované po dobu jedného roka: sieťová aktivita, systémová aktivita, aplikačná aktivita a používateľská aktivita.
- Administrátori musia byť schopní vzdialene monitorovať zachytenú aktivitu a to v reálnom čase.
- Zachytené údaje musia byť ihneď archivované pre budúcu analýzu.
- Záznam musí byť uchovaný pre každý nasadený honeypot.
- Administrátori musia uchovávať štandardizovaný, detailný zápis o každom ohrozenom honeypote.
- Senzory honeynetu musia používať greenwichský čas (Greenwich Mean Time GMT). Jednotlivé honeypoty môžu využívať miestne časové zóny, ale neskôr musia byť tieto údaje konvertované do GMT kvôli ďalšej analýze.

2.5.3 Zber údajov

Väčšina organizácií, ktoré majú nasadený samostatný honeynet, túto funkcionalitu nepotrebuje. Avšak pri viacnásobných honeynetoch je už zber a ukladanie údajov potrebnejšie. Pri korelácii všetkých zozbieraných informácií môžu organizácie zvýšiť výskumnú hodnotu honeynetu.

Prenos zachytených údajov zo senzoru na úložné miesto musí byť bezpečný a zároveň musí byť zabezpečená diskrétnosť, integrita a pravosť údajov. Bezpečnosť je možné zvýšiť aj vytvorením anonymných IP adries organizácie, ale aj ostatných informácií, ktoré je pre organizáciu potrebné držať v tajnosti[10].

2.5.4 Analýza údajov

Tradičné metódy na detekciu podozrivej činnosti sú založené na vybudovaní databázy známych útokov a následnom porovnávaní všetkej prichádzajúcej aktivity s touto databázou. V prípade, že je v databáze nájdená zhoda, je vygenerované upozornenie. Takýto prístup má však aj svoje nedostatky. Prvým nedostatkom je, že doposiaľ neznáme útoky nebudú odhalené. Druhým sú falošné výstrahy. Honeynet je v tomto prípade omnoho efektívnejší nástroj.

Takmer každá prichádzajúca aktivita do systémov honeynetu znamená škodlivú činnosť. Práve preto sú údaje zachytené pri tejto aktivite spoľahlivé a predurčené na analýzu. Vďaka nej je totiž možné odhaliť útoky zamerané na nové, nechránené miesta systému rovnako rýchlo a jednoducho ako aj útoky na už známe nedostatky. Pri analýze údajov zachytených honeynetom a pri hľadaní špecifických vzorov v údajoch môže administrátor určiť nové trendy pri útočení a dokonca predpovedať budúce útoky[12].

3 Vizualizácia v sieťovej bezpečnosti

Z dôvodu potreby analýzy údajov zachytených pomocou honeypotu, resp. honeynetu sa vizualizácia týchto údajov v prehľadnom zobrazení javí ako užitočný prostriedok na rýchle odhalenie nezvyčajnej aktivity v sieti. V tejto kapitole preto rozoberáme všetky aspekty spojené s vizuálnym zobrazovaním údajov.

3.1 Úvod do vizualizácie v sieťovej bezpečnosti

„**Vizualizácia**“ je široký pojem. Vizualizovať znamená vytvoriť mentálny model alebo obraz niečoho. Vizuálne reprezentácie majú dlhú históriu, ale iba pred 20 rokmi sa vizualizácia stala nezávislým výskumným odborom. Cieľom tohto nového odboru je spojiť možnosti ľudského vizuálneho vnímania s výpočtovým výkonom počítačov s cieľom pomôcť používateľom analyzovať a lepšie pochopiť ich údaje, modely, koncepty [1]...Na dosiahnutie týchto cieľov musia byť splnené tri kritériá:

- **Expresivita**- zahŕňa požiadavku zobraziť presne informácie, ktoré sú obsiahnuté v údajoch, teda nič menej a nič viac nesmie byť vizualizované.
- **Efektivita**- stupeň, ako vizualizácia vystihuje poznávacie schopnosti ľudského vizuálneho systému a ďalšie informácie potrebné na dosiahnutie intuitívnej a interpretovateľnej vizuálnej reprezentácie údajov.
- **Vhodnosť a primeranosť**- slúži na posúdenie prínosu vizualizačného procesu s ohľadom na dosiahnutie výsledku danej úlohy.

Pri vizualizačnom procese musíme odpovedať na dve základné otázky:

- **Čo chceme vizualizovať?**- ide o špecifikáciu údajov.
- **Prečo to chceme vizualizovať?**- ide o špecifikáciu úlohy.

3.1.1 Špecifikácia údajov

V posledných rokoch bolo vyvinutých viacero prístupov, ako charakterizovať údaje. Prvý prístup[20] spočíva v rozlíšení závislých a nezávislých premenných. Nezávislé premenné definujú n-rozmerný definičný obor. Hodnoty k závislých premenných sa merajú, resp. definujú množinu k premenných. Ak aspoň jeden rozmer definičného oboru je asociovaný s časom, tak tieto údaje nazývame časovo-orientované údaje.

Ďalší koncept pre modelovanie údajov je **pyramid framework**[14]. Je založený na troch perspektívach:

- **miesto** (kde?),
- **čas** (kedy?),
- **predmet** (čo?).

Čas a miesto charakterizujú definičný obor, t.j. nezávislé premenné. Predmet určuje, čo sa meria, skúma alebo počíta v rámci definičného oboru, t.j. ide o závislú premennú.

3.1.2 Špecifikácia úlohy

Pri špecifikácii toho, čomu má vizualizácia pomôcť potrebujeme rozlíšiť nasledujúce tri ciele:

- Prieskumná analýza
- Potvrdzujúca analýza
- Prezentácia výsledkov analýzy

Pri **prieskumnej analýze** ešte nemáme k dispozícii žiadne hypotézy o daných údajoch. Cieľom je získať celkový pohľad na údaje, extrahovať len relevantné informácie a napokon vytvoriť hypotézy. Vo fáze **potvrdzujúcej analýzy** je vizualizácia použitá na dokázanie alebo vyvrátenie hypotéz. Po týchto dvoch fázach, kedy boli získané fakty o údajoch overené, nasleduje **prezentácia dosiahnutých výsledkov**.

Pri dosahovaní týchto troch základných cieľov vizualizácie sa využívajú dva hlavné koncepty:

- filtrovanie a
- zdôrazňovanie.

Cieľom **filtrovania** je zobraziť len relevantné údaje a tie menej potrebné vynechať. Pri **zdôrazňovaní** sa vyznačujú, zdôrazňujú len tieto potrebné informácie. Pri oboch týchto konceptoch treba byť opatrný, pretože nevieme hneď s istotou určiť, ktoré údaje sú a ktoré nie sú dôležité. Vynechanie alebo zdôraznenie nesprávnych informácii môže viesť k nesprávnej interpretácii vizuálnej reprezentácie[1].

Bertin[3] v roku 1983 popísal, že ľudské vizuálne vnímanie má schopnosť zamerať sa na konkrétny prvok obrázku, na skupinu prvkov, alebo na obrázok ako celok. Na

základe týchto schopností Robertson[15] v roku 1991 vytvoril tri základné kategórie interpretačných cieľov:

- bodový,
- lokálny a
- globálny cieľ.

Indikujú, aké hodnoty nás zaujímajú. Buď ide o hodnoty v danom bode definičného oboru, hodnoty v lokálnej oblasti, alebo hodnoty celého definičného oboru. Tieto základné úlohy môžu byť ďalej delené do viac konkrétnych, špecifických úloh, ktoré sú väčšinou verbálne popísané. Shneiderman[16] v roku 1996 vytvoril taxonómiu úloh, ktorá zahŕňa:

- **Prehľad**- získanie prehľadu o celom datasete.
- **Zoom**- zameranie sa na potrebné údaje.
- **Filter**- odstránenie nepotrebných údajov.
- **Potrebné detaily**- vybrať len potrebné údaje a detaily s nimi spojené.
- **Vzťahy**- vytvoriť pohľad na vzťahy medzi objektmi.
- **História**- udržiavať históriu kvôli možnosti vrátenia sa „o krok späť“.
- **Ukážka**- ukážka údajov a dopytovaných parametrov.

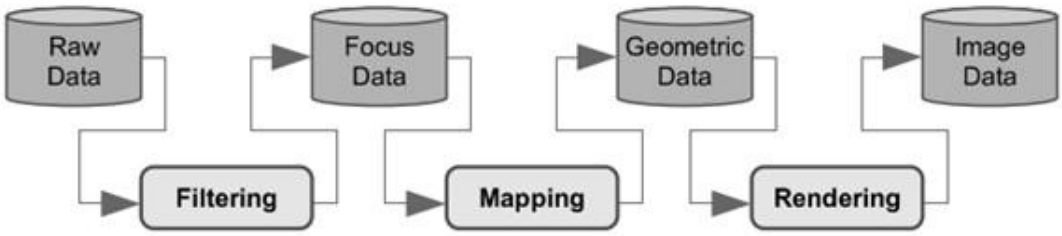
Ako je možné na základe vyššie uvedeného vidieť, úloha vizualizácie môže byť dosiahnutá viacerými metódami. Tie pomáhajú návrhárom vyvíjať reprezentácie, ktoré efektívne podporujú a pomáhajú používateľom pri prieskume a analýze údajov.

3.1.3 Spôsob vizualizácie

Na to, aby sme vytvorili efektívnu vizuálnu reprezentáciu, potrebujeme prvotné, „surové “ údaje namapovať na zodpovedajúce vizuálne atribúty, ako je farba, pozícia, veľkosť, tvar, ktoré spoločne nazývame **vizuálne atribúty**.

Postup pri vizualizácii pozostáva z troch krokov, ktoré je možné vidieť na obrázku č. 2:

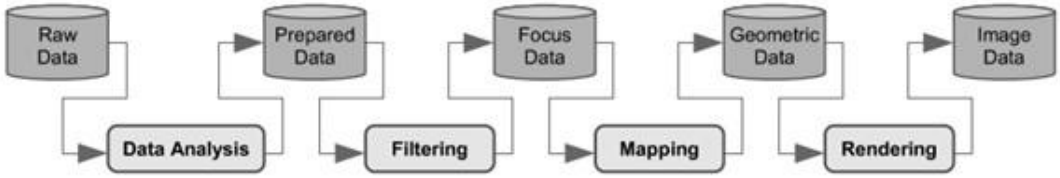
- Filtrovanie
- Mapovanie
- Zobrazenie



Obrázok 2 Postup pri vizualizácii [1]

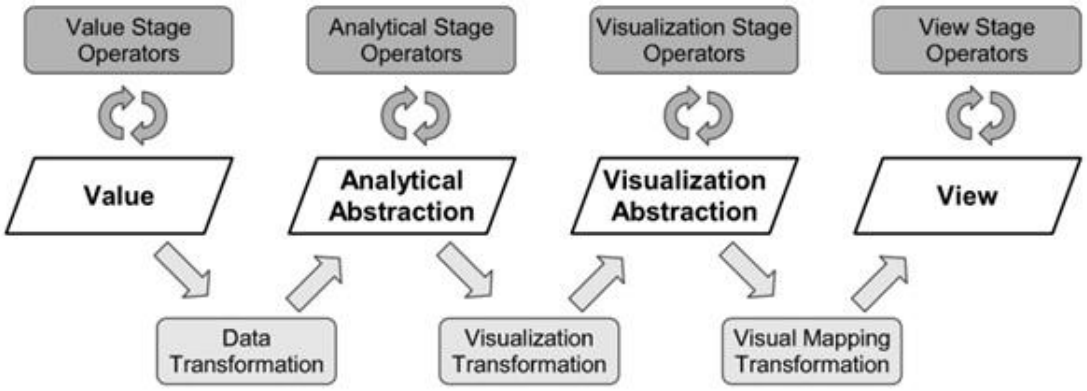
Filtrovanie pripraví vstupné údaje na spracovanie pre nasledujúce dva kroky. Filtrovanie zahŕňa výber relevantných údajov, redukciu údajov, zoskupovanie, redukciu dimenzií a ďalšie. **Mapovanie** má za úlohu namapovať pripravené údaje na príslušné vizuálne premenné. Tento krok je rozhodujúci, pretože vo veľkej miere ovplyvňuje expresivitu a efektivitu výslednej vizuálnej reprezentácie. Nakoniec krok **zobrazenia** generuje zobrazenie z predtým vypočítaných vizuálnych premenných.

V roku 2004 bol vytvorený rozšírený model, v ktorom je krok filtrovania rozdelený na dve samostatné časti: analýza údajov a filtrovanie údajov (obrázok č. 3). Analýza údajov predstavuje automatické výpočty ako zhlukovanie či interpoláciu a filtrovanie extrahuje len tie údaje, ktoré sú pre nás zaujímavé a je potrebné zobraziť ich. Pri viac-rozmerných množinách údajov (-datasetoch) je filtrovanie veľmi dôležité, pretože zobrazenie všetkých informácii by viedlo k ťažko interpretovateľným vizuálnym reprezentáciám[1].



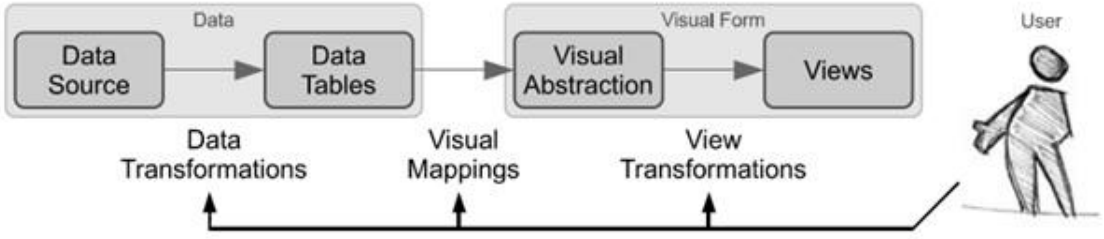
Obrázok 3 Rozšírený model postupu pri vizualizácii [1]

V roku 2000 Chi [6] vytvoril **data state reference model**, teda údajový stavový referenčný model (obrázok č. 4). Tento model reprezentuje postupnú transformáciu abstraktných údajov na obraz cez niekoľko úrovní s využitím operátorov. Operátory spracúvajú údaje na tej istej úrovni abstrakcie.



Obrázok 4 Údajový stavový referenčný model [6]

V roku 1999 bol vytvorený model, v ktorom je používateľ zahrnutý do procesu vizualizácie. Ide o takzvaný **information visualization reference model**, kde používateľ kontroluje vizualizačné spracovanie a interaguje s vizualizačným procesom rôznymi spôsobmi. Rôzne pohľady a obrazy sú vytvárané dovtedy, kým používateľ neskonštatuje, že sú vhodné (obrázok č. 5).



Obrázok 5 Model s používateľom zahrnutým do procesu vizualizácie [5]

3.2 Časovo-orientovaná vizualizácia

Fenomén času bol stále v záujme ľudí. Bolo vyvinutých množstvo teórií pre charakterizáciu fyzickej dimenzie času, ktoré boli tisícky rokov diskutované vo filozofii, matematike, fyzike, astronómii, biológii a mnohých ďalších disciplínach. Vidíme tak, že téma času je veľmi rozsiahla a ľudstvo vynaložilo veľa úsilia na odhalenie jej tajomstiev. Základom všetkých teórii je fakt, že čas je jednosmerný a udeľuje poradie jednotlivým udalostiam.

Prvé náznaky systematického používania nástrojov na zaznamenávanie času boli nájdené vo forme rytín v kostiach, ktoré pripomínajú jednoduché kalendáre založené na cykloch mesiaca. Zložitejšie kalendáre začali vznikať, keď ľudia opustili život lovca a začali sa živiť poľnohospodárstvom. Veľmi dlhú dobu bol čas meraný len približne.

Industrializácia a mestský život však prišli s potrebou presnejšieho, pravidelnejšieho a zosynchronizovanejšieho zaznamenávania času[1].

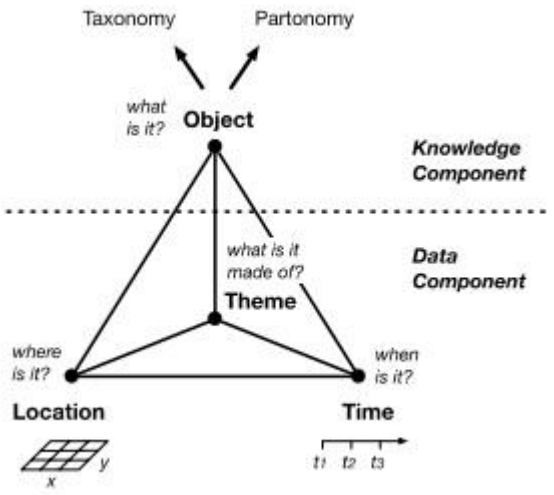
V súčasnosti je najbežnejšie používaným kalendárnym systémom Gregoriánsky kalendár. Bol predstavený pápežom Gregorom VII v roku 1582 s cieľom upraviť predtým používaný Juliánsky kalendár, ktorý bol príliš dlhý v porovnaní s astronomickým rokom a obdobiami. Okrem týchto kalendárnych systémov sa vo svete používajú i ďalšie, ako napríklad Islamský, Čínsky, Židovský kalendár alebo kalendáre pre špeciálne účely, ako napríklad akademické alebo finančné kalendáre [1].

3.2.1 Definícia časovo-orientovaných údajov

Ako sme už spomínali vyššie, ak aspoň jeden rozmer definičného oboru je asociovaný s časom, tak tieto údaje nazývame časovo-orientované údaje[1]. Existuje množstvo prístupov k modelovaniu časovo-orientovaných údajov zahŕňajúce údajové modely od spojitých po diskrétne. Údaje sú väčšinou modelované ako objekty alebo entity, ktoré majú svoje atribúty súvisiace s časom. Užitočný koncept pre modelovanie časovo-orientovaných údajov je **pyramid framework** od Mennisa a ďalších[14] (obrázok č. 6). Tento model je založený na troch perspektívach:

- **umiestnenie**- „kde to je?“,
- **čas**- „kedy to je?“ a
- **samotné údaje**.

Takáto interpretácia údajov vytvára **objekt**- „čo to je?“.



Obrázok 6 Pyramid Framework [14]

3.2.2 Delenie časovo-orientovaných údajov

Na základe domény daných údajov rozlišujeme kvantitatívne a kvalitatívne premenné. **Kvantitatívne premenné** sú založené na diskrétnom alebo spojitom obore, čo umožňuje ich numerické porovnávanie. Naopak **kvalitatívne premenné** zahŕňajú nominálne (neusporiadané) alebo ordinálne (usporiadané) hodnoty. Je dôležité brať do úvahy, o aké údaje podľa tohto kritéria ide, aby bolo možné navrhnúť vhodnú vizuálnu reprezentáciu[1].

Podľa ďalšieho kritéria rozdeľujeme údaje na abstraktné a priestorové. Pod pojmom **abstraktné údaje** rozumieme údajový model, ktorý podľa pyramid framework nezahŕňa aspekt týkajúci sa umiestnenia údajov v priestore. Naopak **priestorové údaje** obsahujú prirodzené priestorové usporiadanie, čiže údajový model zahŕňa aspekt umiestnenia údajov v priestore. Rozdiel medzi abstraktnými a priestorovými údajmi odráža spôsob, akým by mali byť časovo-orientované údaje vizualizované. Pri priestorových údajoch môže byť informácia o priestore využitá na správne namapovanie údajov na obrazovku. Časový aspekt KEDY musí byť začlenený do tohto mapovania. Pri abstraktných údajoch nie je dané žiadne priestorové mapovanie. Preto je predovšetkým potrebné nájsť vhodné výrazné priestorové rozvrhnutie[1].

Podľa toho, o aký druh údajov ide, delíme údaje na údaje vyjadrujúce udalosti a údaje vyjadrujúce stavy. **Udalosti** chápeme ako zmeny stavu (napr. pristávanie lietadla). Na druhej strane **stavy** sú charakterizované ako pretrvávajúce fázy medzi jednotlivými udalosťami (napr. lietadlo je vo vzduchu). Pred vizualizáciou je dobré určiť, o ktorý druh ide, alebo či ide o kombináciu oboch[1].

Podľa počtu premenných rozlišujeme údaje jednorozmerné a viacrozmerné. Ide o počet premenných, ktoré sú závislé na čase. Na reprezentáciu **jednorozmerných** údajov bolo vyvinutých množstvo metód, kým pre **viacrozmerné** údaje je tento počet menší.

3.2.3 Aspekty vizualizácie

Veľa rôznych typov údajov je spojených s časom. Meteorologické údaje, finančné údaje, údaje týkajúce sa sčítania ľudu, údaje z rôznych simulácií alebo plány rôznych projektov obsahujú dočasné informácie. Keďže všetky tieto údaje sú časovo-orientované, teoreticky by ich malo byť možné reprezentovať rovnakým spoločným spôsobom. V praxi sa však tieto typy údajov navzájom odlišujú svojimi špecifickými charakteristikami, a preto každý z nich si vyžaduje im určený spôsob vizualizácie.

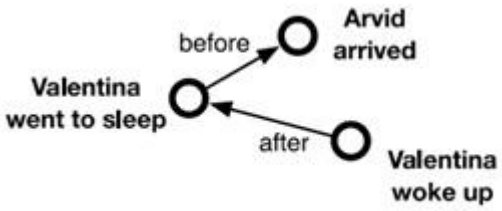
Okrem týchto špecializovaných techník môžu byť časovo-orientované údaje vizualizované tiež s využitím všeobecných prístupov. Keďže čas je prevažne považovaný za kvantitatívnu dimenziu, tak všeobecné vizualizačné frameworky ako Xmdv-Tool [29], Visage [30] či Polaris[31], takisto štandardné vizualizačné techniky ako paralelné súradnice alebo viac či menej sofistikované diagramy a grafy sú aplikovateľné na vizualizáciu časovo-orientovaných údajov[1].

Existuje množstvo konceptov pre analýzu časovo-orientovaných údajov, ktoré vieme nájsť v množstve literatúry. Toto množstvo rôznych prístupov tak sťažuje výber najvhodnejšieho prístupu pre dané údaje.

3.2.4 Príklady údajov a ich vizualizácia

Na to, aby sme dokázali určiť najvhodnejšiu metódu pre vizualizáciu konkrétnych údajov, musíme sa pozrieť na aspekty, ktoré charakterizujú rôzne typy času.

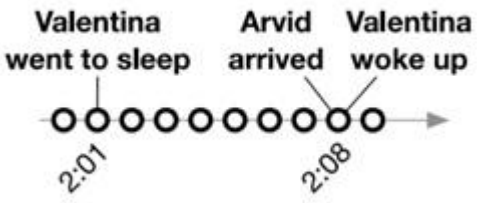
Typy časov môžeme rozdeliť na diskrétne, spojité a ordinálne. V ordinálnej časovej doméne je zobrazované iba relatívne poradie vzťahov jednotlivých objektov, resp. udalostí teda vieme určiť, čo sa stalo pred alebo po nejakej inej udalosti. Pomocou takéhoto typu času vieme modelovať napríklad situáciu, že Valentína šla spať pred tým než prišiel Arvid a Valentína vstala po pár minútach spánku ako to zobrazuje nasledujúci obrázok č. 7 [1].



Obrázok 7 Ordinálna časová doména [1]

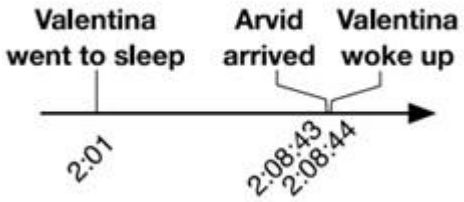
Takýto spôsob vizualizácie môžeme použiť len v prípade, ak nás zaujímajú len kvalitatívne dočasné vzťahy a k dispozícii nemáme žiadne kvantitatívne informácie.

V diskrétnych časových doménach sa časové hodnoty mapujú na množinu celých čísel. Takéto časové domény sú založené na najmenšej možnej jednotke (napríklad sekundy alebo milisekundy) a sú to najčastejšie používané časové modely v informačných systémoch[1].



Obrázok 8 Diskrétna časová doména [1]

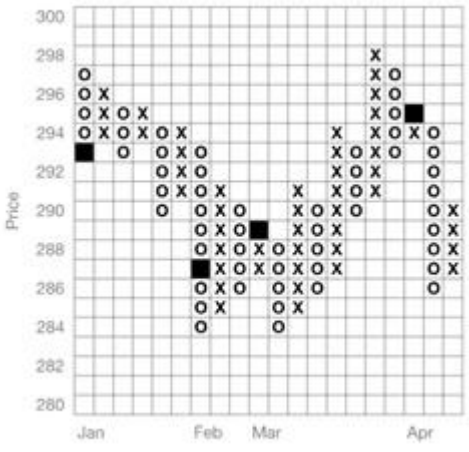
Spojité časové modely sú charakterizované mapovaním na reálne čísla, to znamená, že medzi dvoma časovými bodmi existuje ďalší bod[1].



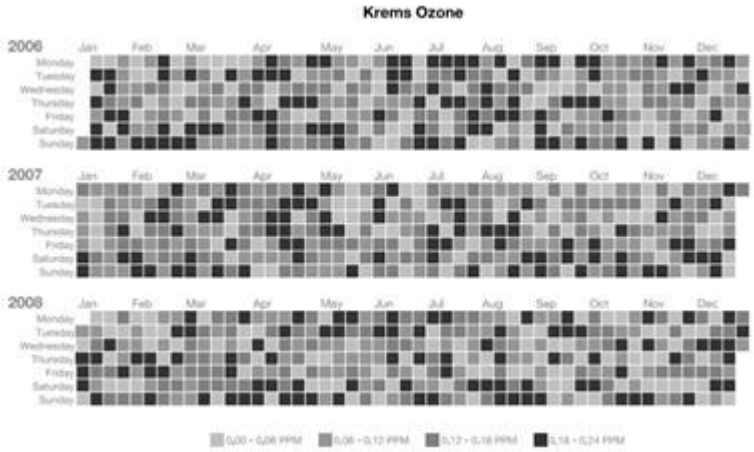
Obrázok 9 Spojitá časová doména [1]

Príklady vizualizačných techník, ktoré umožňujú reprezentovanie týchto troch typov časových domén je

- **point and figure chart** pre ordinálnu (obrázok č. 10),
- **tile maps** pre diskrétnu (obrázok č. 11) a
- **circular silhouette graph** pre spojitú časovú doménu (obrázok č. 12).



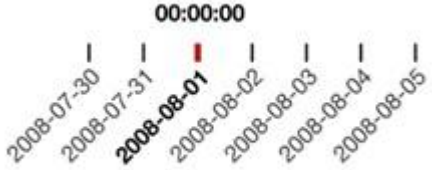
Obrázok 10 Point and Figure Chart [1]



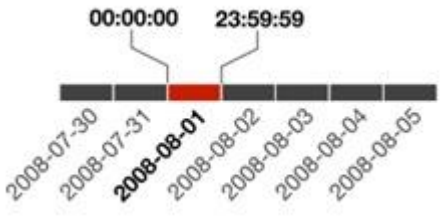
Obrázok 11 Tile Map [1]

Obrázok 12 Circular Silhouette Graph [1]

Podľa štruktúry časovej domény rozlišujeme **časovú doménu založenú na bodoch** (obrázok č. 13), kde nie je daná žiadna informácia o intervale medzi dvoma bodmi. Naopak **časová doména založená na intervaloch** (obrázok č. 14) sa týka jednotlivých podintervalov času s dĺžkou väčšou než nula. Napríklad časová hodnota 1. August 2008 sa môže týkať jediného okamihu 1. August 2008 00:00:00 v časovej doméne založenej na bodoch, kým v časovej doméne založenej na intervale ide o celý interval [1. August 2008 00:00:00, 1. August 2008 23:59:59] [1].

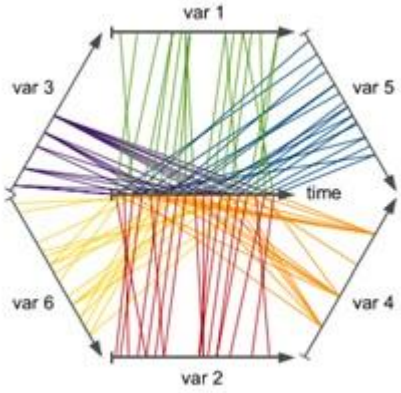


Obrázok 13 Časová doména založená na bodoch [1]



Obrázok 14 Časová doména založená na intervaloch [1]

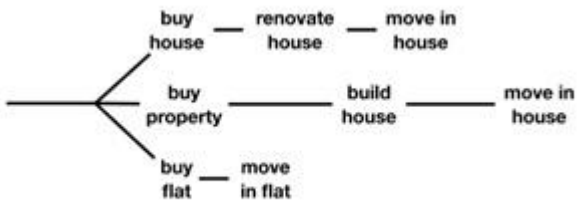
Príklady vizualizačných techník, ktoré umožňujú reprezentovať tieto dva typy časových domén sú **TimeWheel** (obrázok č. 15) pre časovú doménu založenú na bodoch a **tile maps** pre časovú doménu založenú na intervale.



Obrázok 15 TimeWheel [1]

Podľa usporiadania časovej domény rozlišujeme čas idúci **lineárne** z minulosti do budúcnosti, to znamená, že každá časová hodnota má jednoznačného predchodcu i nasledovníka. V **cyklickej** organizácii času je celá doména času zložená z množiny opakujúcich sa časových hodnôt (napríklad ročné obdobia). V tomto prípade je každá hodnota A predchádzaná, ale aj nasledovaná ktoroukoľvek inou hodnotou B v tom istom čase- napríklad zima je pred letom, ale aj za letom. Príkladom vizualizačnej techniky pre lineárny čas je **TimeWheel** [36] a pre cyklický **circular silhouette graph**.

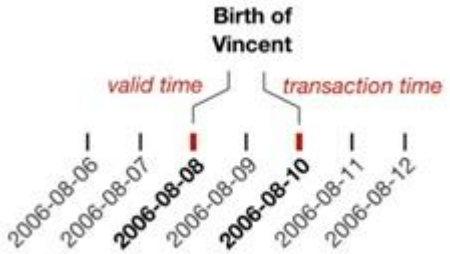
Podľa ďalšieho kritéria rozlišujeme **usporiadanú časovú doménu**, v ktorej sa udalosti dejú jedna po druhej. Na detailnejšej úrovni tiež môžeme rozlíšiť **úplne** a **čiastočne usporiadanú časovú doménu.** V úplne usporiadanej doméne sa v jednom časovom okamihu môže stať iba jedna udalosť. Na rozdiel od toho v čiastočne usporiadaných časových doménach sú povolené aj simultánne a prekrývajúce sa udalosti. Zložitejšia forma organizácie časovej domény je takzvaný **branching time** (obrázok č. 16)[1].



Obrázok 16 Branching Time [1]

Ako je vidieť na obrázku č. 16, ide o viacero vetiev, ktoré umožňujú popis a porovnávanie alternatívnych scenárov, napríklad v plánovaní projektu. Tento typ časovej domény podporuje rozhodovací proces, kde iba jeden zo zobrazených scenárov sa skutočne aj uskutoční. Tento spôsob nie je užitočný len pre budúce scenáre, ale i pre vyšetrovanie minulosti. Na rozdiel od branching time, kde sa uskutoční iba jedna vetva, **viacnásobné perspektívy (multiple perspectives)** umožňujú simultánny pohľad na čas, čo je potrebné napríklad na vytvorenie štruktúry výpovedí svedkov[1].

V časových databázach sú často modelované dve perspektívy- platný čas a čas transakcie (obrázok č. 17). Platný čas je čas, kedy je daný fakt pravdivý v modelovanej realite - napríklad Vincent sa narodil 8. augusta 2006. Čas transakcie predstavuje čas, kedy bol údaj uložený do databázy.



Obrázok 17 Platný čas a čas transakcie [1]

Branching time aj multiple perspectives predstavujú potrebu vysporiadať sa s pravdepodobnosťou, resp. neurčitosťou pri určovaní, ktorá vetva v čase sa s najväčšou pravdepodobnosťou uskutoční alebo napríklad ktoré svedectvo je najhodnovernejšie[1]. Príkladom vizualizačnej techniky umožňujúcej reprezentáciu branching time je **decision chart**.

3.3 Vizualizácia časovo-orientovaných údajov získaných pomocou honeypotov a honeynetov

Na to, aby sme vedeli určiť, ktoré konkrétne údaje budeme vizualizovať, musíme vedieť, ako sú časovo-orientované údaje definované a na základe toho vybrať len tie údaje, ktoré tejto definícii zodpovedajú. Keď hovoríme o časovo-orientovaných údajoch, tak myslíme tie údaje, ktoré sú nejakým spôsobom spojené s časom [2]. Ďalšia možnosť ako definovať časovo-orientované údaje je, že ide o údaje, v ktorých zmeny v čase alebo rôzne dočasné aspekty hrajú podstatnú rolu, respektíve sú v našom záujme [21].

Otázkou je, aké údaje dokážeme pomocou honeypotov a honeynetov získavať. Existujúce riešenia vizualizácií údajov získaných pomocou honeypotu, resp. honeynetu, zobrazujú napríklad nasledovné informácie:

1. najčastejšie skúšané heslá na získanie prístupu do systému
2. najčastejšie skúšané používateľské mená na získanie prístupu do systému
3. najčastejšie skúšané kombinácie hesiel a používateľských mien na získanie prístupu do systému
4. celkový počet úspešných prístupov do systému
5. počet úspešných prihlásení za deň
6. počet úspešných prihlásení za týždeň
7. počet vytvorených spojení z konkrétnej IP adresy
8. počet úspešných prihlásení z konkrétnej IP adresy
9. počet pokusov o prístup do systému za deň
10. počet pokusov o prístup do systému za týždeň
11. top 10 SSH klientov
12. počet vytvorených spojení z krajiny
13. počet vytvorených spojení z konkrétnej IP adresy spolu s kódom krajiny
14. dni s najvyššou aktivitou potenciálnych útočníkov
15. najčastejšie zadávané vstupné príkazy
16. počet úspešných vstupných príkazov
17. počet neúspešných vstupných príkazov

V našej práci sa však zaoberáme časovo-orientovanými údajmi, čím sa snažíme priniesť nový pohľad na údaje získané pomocou honeynetu. Vizualizácie založené na

čase tak môžu priniesť nové poznatky o útokoch či postupe útočníka. Pri pripojení na službu honeypotu, resp. honeynetu sa do databázy automaticky vkladá časová pečiatka súvisiaca s daným pripojením. Práve preto všetky takéto údaje sú podľa definície časovo-orientované a vieme z nich vyčítať nasledovné zaujímavé informácie:

1. Vieme zobraziť priebeh počtu pripojení na honeynet za určitý časový úsek, t.j. pri pohľade na graf vieme určiť koľko útokov bolo zaznamenaných v konkrétny deň, hodinu či mesiac.
2. Keďže máme k dispozícii časovú pečiatku, vieme vytvoriť štatistiku napríklad na konci každého mesiaca, z ktorej bude jasné koľko pripojení bolo zaznamenaných na jednotlivých službách.
3. Keďže sa z IP adresy dá zistiť, o ktorú časovú zónu ide, ponúka sa možnosť zobraziť počty pripojení z jednotlivých časových zón.

Časovo-orientované údaje majú väčšinou veľký objem. Jednoducho povedané, je ich veľa - a to nie len v zmysle počtu údajových položiek, ale tiež v zmysle počtu sledovaných atribútov. Obyčajné, nepremyslené vizualizácie takýchto údajov môžu viesť k neprehľadným, preplneným reprezentáciám, čo robí takéto vizualizácie nepoužiteľnými. Avšak vhodne zvolené vizualizačné techniky pomáhajú získať celkový pohľad aj na veľké údajové súbory [2]. V nasledujúcom texte sa preto bližšie pozrieme na niekoľko konkrétnych techník, ktoré je možné využiť na vizualizáciu údajov získaných pomocou honeypotu.

4 Aktuálne prístupy k vizualizácii v sieťovej bezpečnosti

V súčasnosti existuje množstvo vizualizačných prostredí a rôznych knižníc na zobrazovanie údajov. V tejto kapitole je vytvorený prehľad rôznych riešení vizualizácií údajov z bezpečnostných systémov. Špeciálne sa zameriavame na vizualizácie údajov z honeypotov. Keďže sú časovo-orientované údaje špeciálnym typom údajov, ktorým sa venujeme v tejto práci, venujeme poslednú podkapitolu existujúcim prístupom k zobrazovaniu práve týchto údajov.

4.1 Vizualizácia údajov z honeypotov

Keďže vďaka honeypotom získavame veľké množstvo údajov, tak je dôležité vedieť sa v nich zorientovať, získať prehľad kvôli ďalšiemu spracovaniu a analýze. Vhodnou vizualizáciou týchto údajov je možné tento cieľ dosiahnuť.

Vizualizácia súvisiaca s **SSH honeypotmi** bola vytvorená Jamie Blascom[4]. Ide o vizualizáciu údajov získaných Nephentes honeypotom. Tento honeypot sa využíva na získavanie vzoriek malwaru emulovaním nechránených, napadnuteľných miest systému Windows, ktoré potenciálny útočník využije a pošle malware. Táto vizualizácia mapuje zdrojovú IP adresu útočníka na krajinu odkiaľ je útok zaznamenaný. Takto je potom hneď viditeľné, z ktorej krajiny je daný malware rozširovaný. Napriek tomu však informácia o krajine nemusí byť pravdivá, keďže útočníci často využívajú proxy servery s cieľom utajiť svoju skutočnú pozíciu. Vizualizácia teda mapuje len proxy servery, nie skutočný zdroj útoku.

Pri súbore viacerých honeypotov je spravovanie všetkých záznamov o útokoch zložitejšie. V roku 2011 Visoottiveseth a iní[19] vytvorili architektúru na centrálne ukladanie týchto záznamov z viacerých honeypotov. V ich experimentoch sa však okrem vizualizácie údajov žiadna ďalšia analýza týchto údajov nerobila.

Kippo-graph [23] je skript určený na vytváranie základných vizualizácií údajov získaných z SSH honeypotu Kippo. Takisto ako pri predchádzajúcom, ani tu nie je povolená žiadna interakcia s údajmi. Využíva **Libchart PHP** knižnicu od Jean-Marca Trémeauxa**, QGoogleVisualizationAPI PHP Wrapper for Google's Visualization API** od Thomasa Schäfera, **RedBeanPHP** knižnicu, ktorej autorom je Gabor de Mooij a geolokačnú technológiu od **geoPlugin**. Kippo-graph v súčasnosti zobrazuje 24 grafov, medzi ktorými je napríklad graf zobrazujúci top 10 hesiel, ktoré útočníci skúšajú pri

napadnutí systému, top 10 používateľských mien, top 10 kombinácií mena a hesla, počet pokusov za deň či týždeň, úspešné či neúspešné zadané vstupy a tak ďalej. Súčasťou sú tiež geolokačné údaje extrahované a zobrazené pomocou Google vizualizačnej technológie využívajúcej Google Map, Intensity Map a tak ďalej. V neposlednom rade dokáže Kippo-graph zobraziť takzvaný tty log, teda do textového súboru ukladá postupnosť krokov útočníka.

Vizualizácii údajov z honeypotov s využitím nástrojov **Graphviz** a skriptov **Afterglow** [24] sa venoval vo svojej práci z roku 2009 Craig Valli. Práca študuje použitie Graphvizu a Afterglowu na analýzu údajov z honeypotu. Údaje získané honeypotom sú extrahované a spracované Afterglow skriptami pričom vznikajú výstupy, ktoré sú vhodné pre nástroje zahrnuté v Graphvize.

Ďalší nástroj na správu údajov z viacerých honeypotov je **SURFcert IDS** [25], open-source Distributed Intrusion Detection System, ktorý je založený na pasívnych senzoroch. Všetky záznamy môžu byť analyzované s využitím webového rozhrania. Avšak žiadna interakcia s údajmi nie je možná.

Zaujímavým spôsobom sú vizualizované údaje získané vďaka senzorom Honeynet Project-u na takzvanej **Honeymape** [26]. Ide o mapu sveta, kde je v reálnom čase možné vidieť odkiaľ, z akej krajiny je vedený útok na niektorý z honeypotov Hoenynet Project-u. Ako samotní autori hovoria, Honeymap vznikla skôr pre zábavu, autori nemali žiadny vážny cieľ. Výsledok je naozaj zaujímavý a užitočný. Autori Florian Weingarten a Mark Schloesser z univerzity v Aachene tvrdia, že išlo len o vytvorenie niečoho pekne vyzerajúceho a využívajúceho technológie ako CSS3 animácie alebo HTML5 webové sokety.

4.2 Vizualizácia údajov z bezpečnostných systémov

O vizualizácii útokov na počítačové siete sa robí množstvo výskumov, avšak najviac pozornosti sa venuje vizualizácii útokov, ktoré sú zaznamenávané prostredníctvom Intrusion Detection System (IDS) v kombinácii s NetFlow údajmi (ide o údaje zozbierané vďaka NetFlow sieťovému protokolu vyvinutému na monitorovanie prevádzky v sieti). Nástroj **NFlowVis** prezentovaný v roku 2008 Fischerom a inými [8] sa dá využiť na vizuálnu analýzu útokov v rozsiahlych sieťach. Tento nástroj vizualizuje NetFlow z útokov, ktoré boli zaznamenané pomocou IDS. Zobrazené sú len spojenia na SSH server.

Ďalší nástroj na vizualizáciu NetFlow údajov je **VIAssist** vyvinutý v roku 2009 Goodallom a ďalšími [9]. Tento nástroj zobrazuje NetFlow údaje na veľkej prístrojovej doske a expert tak môže jednoducho vyhľadávať v údajoch s využitím rôznych pohľadov. Je možné na žiadosť zobraziť detaily útoku, v prípade keď je niektorý tok údajov potrebné analyzovať do hĺbky. VIAssist vizualizuje len NetFlow údaje a nemôže byť použitý pre analýzu útokov na SSH honeypoty, keďže chceme nájsť možné súvislosti medzi jednotlivými útokmi a to na základe obsahu každého útoku.

Pravail Security Analytic [27] umožňuje interagovať s údajmi vďaka vizualizácii, ktorá zobrazuje údaje z viacerých perspektív či už ide o útočníka, cieľ útoku alebo typ útoku. Takisto umožňuje rýchlo porovnávať štatistiky z jednotlivých útokov, ktoré sa udiali z rôznych miest v rôznom čase.

Gephi [28] je open-sourcová interaktívna vizualizačná a prieskumná platforma pre všetky typy sietí a komplexných systémov, dynamické a hierarchické grafy. Používateľ interaguje s vizuálnou reprezentáciou tak, že manipuluje so štruktúrami, tvarmi, farbami a tak odhaľuje charakteristiky daných údajov. Cieľom tohto nástroja je pomôcť analytikovi vytvoriť hypotézu o údajoch, s ktorými pracuje.

4.3 Vizualizácia časovo-orientovaných údajov

Na zobrazenie časovo-orientovaných údajov existujú techniky, ktoré po aplikácii na údaje vytvárajú prehľadnú vizuálnu reprezentáciu týchto údajov. Asi najznámejší existujúci spôsob zobrazenia údajov je karteziánsky súradnicový systém, kde na horizontálnej osi vidíme čas a na vertikálnej zodpovedajúce hodnoty. Napríklad v **Point Plot** zobrazení sú údaje zobrazené ako body, v **Line Plot** zobrazení ako krivky, ďalej poznáme typické stĺpcové grafy a mnoho ďalších techník na zobrazenie údajov. Niektoré z nich si hneď aj predstavíme.

Jedným z najčastejšie používaných spôsobov zobrazovania časovo-orientovaných údajov je spomínaný **Line Plot** [1]**.** Ide o rozšírenie **Point Plot-u** a to tým, že jednotlivé údajové body sú spojené čiarami, čím sa zdôrazňuje závislosť na čase.

V dôsledku toho sa **Line Plot** zameriava na celkový tvar údajov v priebehu času. Medzi údajovými bodmi sa môžu vyskytovať rôzne štýly čiar, napríklad úsečky, prerušované čiary či krivky. Dôležité je si uvedomiť, že spojenia medzi jednotlivými bodmi sú len aproximáciou, a preto hodnoty v časovom intervale medzi dvoma

údajovými bodmi nemusia byť presné. Existuje množstvo rozšírení tohto grafu, a to napríklad **Fever Graph**, ktorý sa používa na zobrazenie údajov, ktorých hodnoty sa menia spojite, napríklad ceny akcií, ďalej kontrolné grafy, indexové grafy a podobne.

TimeTree [5] od Carda a ďalších autorov je vizualizačná technika umožňujúca prieskum meniacich sa hierarchických organizovaných štruktúr a jednotlivých súčastí týchto štruktúr. Takáto vizualizácia pozostáva z 3 častí:

- **časový bežec** (slider) na spodnej časti,
- **samotný stromový graf** v strede a
- **pole pre vyhľadávanie** na vrchu tejto reprezentácie.

Vďaka časovému bežcu (slideru) je možné pristúpiť ku ktorémukoľvek časovému okamihu. Táto stromová vizualizácia používa „degree-of-interest" prístup (DOI) kvôli zvýrazneniu dôležitých informácií. Preto je použitá špecifická farebná schéma na určenie charakteristík v údajoch, ako sú napríklad dôležité uzly, vyhľadávané uzly či uzly, na ktoré sa naposledy kliklo.

Erbacher a iní popísal systém, ktorý vizualizuje záznamy automaticky generované monitorovaným systémom. Vizualizácia **Intrusion Monitoring** [7] zobrazuje monitorovaný systém ako bod v strede a zobrazuje počet používateľov a zaťaženie servera. Udalosti sú zobrazené ako do stredu smerujúce čiary, na konci ktorých je zobrazený vzdialený stroj (host) ako malý bod. Bežné aktivity v sieti majú sivú farbu pričom nečakané, podozrivé aktivity majú odlišnú, výraznejšiu farbu, napríklad neoprávnené vniknutie do systému je zobrazené červenou farbou. Takýto typ vizuálnej reprezentácie pomáha administrátorovi sledovať sieťovú komunikáciu.

Ak potrebujeme zobraziť nie len jednotlivé body v čase, ale zaujímame sa o celé intervaly, teda ako dlho nejaká udalosť trvala, je výhodné použiť **Timeline** [1]. Udalosť a jej trvanie sa zobrazí čiarou alebo úzkym obdĺžnikom paralelne s časovou osou. Takýto spôsob vizualizácie je veľmi účinný a bol využívaný ešte predtým než sa objavili prvé počítače. Príkladmi takýchto časových osí sú napríklad **LifeLine** alebo **Gantt charts.**

Rodopisné údaje sú veľmi zaujímavým zdrojom časovo-orientovaných informácií. V takýchto údajoch nájdeme nie len štruktúru rodiny, ale aj vzťahy medzi jednotlivými členmi. **TimeNet** [1] reprezentuje každú osobu ako samostatný pás umiestnený horizontálne pozdĺž časovej osi zľava doprava. Na každom páse je napísané

meno osoby a farba pásu označuje pohlavie- červená je vyhradená pre ženy a modrá pre mužov. Manželstvo dvoch osôb je znázornené priblížením ich pásov a rozvod oddialením týchto pásov. Pri narodení dieťaťa je od rodičov vedená smerom dole prerušovaná čiara, z ktorej horizontálne s časovou osou vychádza nový pás reprezentujúci toto dieťa. Vo vizualizácii nájdeme aj označenie historických udalostí či osobných udalostí členov rodiny.

5 Vizualizácia priebehu útoku na honeynet

V tejto kapitole sa venujeme výberu vizualizačných metód vhodných pre časovo-orientované údaje. Analyzujeme vizualizačné prostredia, ktoré sú pre vybrané metódy využiteľné a následne popisujeme, akým spôsobom je možné vybrané metódy využiť pre zobrazovanie časovo-orientovaných dát z honeynetu a ako sme postupovali pri implementácii. V závere každej podkapitoly týkajúcej sa jednotlivých metód zhŕňame výsledky, t.j. čo implementované metódy prezrádzajú o priebehoch útokov na honeynet.

5.1 Príprava údajov

Údaje, ktoré vizualizujeme vybranými vizualizačnými metódami popísanými v nasledujúcich kapitolách boli zozbierané z honeynetu umiestneného na Masarykovej univerzite v Brne, kde sú nasadené aj vysoko-interaktívne, aj nízko-interaktívne honeypoty. Základ honeynetu tvorí nízko-interaktívny honeypot HoneyD. V databáze s údajmi, ktoré tento honeynet zachytil sa nachádza vyše 4 miliónov záznamov o pripájaní sa na služby SSH, POP3 a IMAP.

Pôvodná databáza obsahovala tabuľku so stĺpcami IP adresa, služba a časová pečiatka, ktorá odpovedá času, kedy bol zaznamenaný pokus o pripojenie k danej službe. Pre potreby našej vizualizácie bola tabuľka rozšírená pomocou dvoch PHP skriptov o dva stĺpce: o miestnu časovú zónu a GMT časovú zónu. Oba tieto údaje bolo možné zistiť na základe známej IP adresy pomocou systému IP-API [39].

Pri vizualizácii s názvom Heatmap sme využili aj druhú tabuľku tejto databázy. V tejto tabuľke sú uvedené záznamy o miestach, z ktorých bolo zaznamenané pripojenie na honeynet. Máme tak k dispozícii stĺpce IP adresa, krajina, mesto, poskytovateľ internetového pripojenia (ISP). Ostatné stĺpce tejto tabuľky v práci nevyužívame.

5.2 Výber vizualizačných metód využiteľných pre vizualizáciu

Pre našu prácu sme zvolili tri vizualizačné metódy popísané nižšie, ktorým sa podrobnejšie budeme venovať v nasledujúcich podkapitolách:

Line Plot – aj keď sa táto vizualizačná metóda javí ako klasický graf s dvoma osami a krivkou, my sme ho upravili a doplnili doň „farebné balóniky“, ktoré slúžia na zobrazenie ďalšej, dodatočnej informácie, v našom prípade zobrazujú mesačnú štatistiku počtu pripojení na jednotlivé služby honeynetu. Takéto zobrazenie je potom prehľadné aj napriek množstvu informácií, ktoré zobrazuje. Pomocou bežca vieme

zobrazovať údaje pre jednotlivé mesiace v roku a sledovať tak priebeh počtu pripojení detailnejšie ako pri súhrnnom grafe pre celý rok.

Heatmap - Výhodou heatmapy je, že v 2D priestore dokážeme zobraziť nielen 2 atribúty ako je to v prípade klasického grafu s dvoma osami a krivkou, ale v matici vieme pomocou farby každého políčka a prislúchajúcej legendy zobraziť aj tretiu hodnotu.

Treemap - je vizualizácia určená pre hierarchické dáta, čo je výhodné pre zobrazovanie jednotlivých časových pásem, keďže do jednej GMT zóny patrí viacero miestnych časových zón, čo vytvára hierarchiu v dátach. Okrem toho vieme využiť nielen farbu, ale aj veľkosť políčok na zakódovanie jednotlivých atribútov. Na zachytenie priebehu počtu pripojení z jednotlivých pásem je možné využiť bežec (slider), posúvaním ktorého sa budú zobrazovať len údaje pre aktuálne jazdcom vybratý mesiac.

5.3 Analýza vizualizačných prostredí (frameworkov) využiteľných pre zvolené vizualizačné metódy

V súčasnosti existuje viacero frameworkov pre zobrazovanie údajov, a preto bolo potrebné zvoliť ten, ktorý sa pre zvolené vizualizačné metódy hodí najviac.

D3.js je JavaScriptová knižnica určená pre manipuláciu s údajmi. Pri zobrazovaní údajov pomocou tejto knižnice je možné využiť nielen JavaScript, ale takisto HTML, CSS či SVG. Kombinuje silné vizualizačné komponenty s údajovo orientovaným prístupom a umožňuje tak vytvárať moderné a interaktívne vizualizácie.

HighCharts je knižnica napísaná v JavaScripte, ktorá poskytuje jednoduchý spôsob pridávania interaktívnych grafov na webové stránky či do webových aplikácií. Nevýhodou však je, že v súčasnosti podporuje len vytváranie čiarových, stĺpcových, koláčových a im podobných grafov, čím je pre naše zvolené vizualizačné metódy nepoužiteľná.

JSCharts je takisto knižnica založená na JavaScripte. Umožňuje jednoduché generovanie grafov a nevyžaduje žiadne dodatočné pluginy či moduly. Dokáže spracovať údaje vo formátoch XML, JSON aj z JavaScriptového poľa. Pre naše vizualizačné metódy je však z podobného dôvodu ako HighCharts nepoužiteľná- zameriava sa totiž na grafy zobrazujúce klasické štatistické údaje, ako je stĺpcový či koláčový graf.

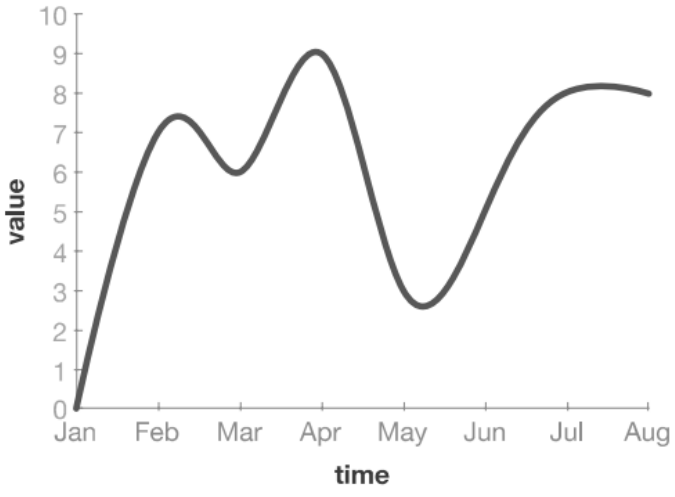
FusionCharts je komplexná JavaScriptová knižnica, ktorá poskytuje dobre vyzerajúce a vysoko interaktívne vizualizácie. Vyznačuje sa dobre spracovanou dokumentáciou, množstvom príkladov a návodov, ktoré ju robia ľahko použiteľnou. V našej práci sme sa však nerozhodli ani pre túto knižnicu, pretože je potrebné zakúpiť si licenciu na jej používanie.

Z vyššie spomínaných dôvodov sme sa rozhodli pri implementácii využiť práve knižnicu d3.js, ktorá podporuje „moderné“ prehliadače ako Firefox, Chrome, Safari, Opera, IE9+, Android aj iOS.

5.4 Line Plot

Line Plot je jedna z najčastejších reprezentácií časovo-orientovaných údajov. Ide o rozšírenie bodových grafov, čo znamená, že jednotlivé body sú pospájané krivkou, čo zvýrazňuje časovú povahu údajov. Tento spôsob vizualizácie sa zameriava na celkový tvar údajov v čase. Treba si však uvedomiť, že krivka vykreslená medzi jednotlivými bodmi nemusí zodpovedať skutočnosti, pretože ide len o aproximáciu či už priamkami alebo Bézierovou krivkou.

Existuje viacero rozšírení Line Plotu, medzi ktoré patria napríklad kontrolné, indexové grafy alebo skupinové grafy, kde vieme zobraziť minimálnu, strednú a maximálnu hodnotu v každom okamihu.



Obrázok 18 Line Plot [1]

5.4.1 Využitie v honeynete

Táto vizualizačná technika doplnená ďalším prvkom popísaným nižšie je vhodná pre zobrazovanie údajov v honeynete nasledujúcim spôsobom: krivka vo

všeobecnosti zobrazuje počet spojení s honeynetom počas časového intervalu v závislosti od údajov. Z grafu teda vidíme, ako sa v čase menil počet spojení s honeynetom. Keďže sa na honeynet útočí z rôznych IP adries, je možné zobraziť graf priebehu počtu spojení z konkrétnej vybranej IP adresy. Ten, kto využíva takúto vizualizáciu sa tak môže zamerať len na konkrétnu IP adresu a sledovať aktivitu prichádzajúcu z nej.

Prvok, ktorý túto techniku ozvláštňuje a umožňuje zobraziť pridané informácie, sú, nazvime ich, farebné „balóniky“. Tie vieme využiť tým spôsobom, že v rámci nejakého časového intervalu, napríklad jedného mesiaca, vieme zobraziť „balónik“, vnútri ktorého je informácia o tom, koľko útokov, resp. spojení bolo v danom mesiaci zaznamenaných na konkrétnu službu. Pri analýze jednotlivých mesiacov tak vieme hneď povedať, s ktorou službou bolo vytvorených najviac spojení.

Už samotný LinePlot zobrazuje priebeh počtu pripojení na honeynet. Pomocou bežca (slidera) je však možné zdokonaliť zobrazovanie priebehu počtu pripojení tým spôsobom, že pri jeho posúvaní pozdĺž osi, na ktorej sú nanesené mesiace roka, sa graf prekresľuje a zobrazuje len údaje pre zvolený mesiac.

5.4.2 Implementácia

Pri implementácii sme využili JavaScriptovú knižnicu d3.js. Naša MySQL databáza obsahuje jednu tabuľku s piatimi stĺpcami predstavujúcimi ip adresu, časovú pečiatku, miestnu časovú zónu, GMT a službu. Na to, aby sme potrebné údaje vytiahli z databázy a zobrazili ich, sme využili PHP skript s nasledujúcim dopytom. Tento dopyt je vytvorený tak, aby sa v zobrazených údajoch nevyskytovala maximálna ani minimálna hodnota, teda takzvané „outliers“, ktoré môžu predstavovať akúsi chybu. Napríklad, ak sa v údajoch vyskytne hodnota 0 pripojení v nejaký deň, tak to môže indikovať, že v ten deň bol systém nefunkčný. Ak by sme túto hodnotu reálne interpretovali ako 0 pripojení v daný deň, mohlo by to celú vizualizáciu skresliť.

```
select * from (select date(timestamp) as date, count(timestamp) as pct50 from `dataset_complete`where ip='".$arg. "'and month(timestamp)='".$arg2. "' group by date(timestamp)) t1 where t1.pct50!=(select max(pct50) as max from (select date(timestamp) as date, count(timestamp) as pct50 from `dataset_complete` where ip='".$arg. "' and month(timestamp)='".$arg2. "' group by date(timestamp)) t1) and t1.pct50!=(select min(pct50) as max from (select date(timestamp) as date,
```

count(timestamp) as pct50 from `dataset_complete` where ip='".$arg. "' and month(timestamp)='".$arg2. "'group by date(timestamp)) t1)

Keďže okrem celkového prehľadu pre celý honeynet chceme aj údaje iba pre konkrétnu zvolenú IP adresu, museli sme hodnotu zvolenú v comboboxe „poslať“ v URL adrese nášmu PHP skriptu. Túto hodnotu v predchádzajúcom dopyte predstavuje premenná $arg. PHP skript vytvoril JSON, ktorý sme následne v JavaScripte načítali, naparsovali a získané dáta vizualizovali s využitím d3.js.

Ďalší problém, ktorý bolo potrebné vyriešiť, bolo naplnenie comboboxu IP adresami, ktoré sa nachádzajú v tabuľke. Na ich získanie sme využili jednoduchý PHP skript s dopytom:

select distinct(ip) from `dataset_complete`

Získaný JSON sme pretransformovali na pole a vo FOR cykle sme hodnoty z tohto poľa vložili do comboboxu.

Ako bolo spomínané vyššie, balóniky môžu predstavovať akúsi štatistiku na konci každého mesiaca. Preto bolo potrebné vytvoriť dopyt, ktorý zabezpečí umiestnenie balónika na koniec každého mesiaca a zistí informáciu o tom, koľko pripojení bolo za daný mesiac zaznamenaných na jednotlivé služby.

select last_day(timestamp) as date, count(service) as version, service, service as type, count(service)as count from `dataset_complete` where ip='".$arg. "' group by last_day(timestamp), service

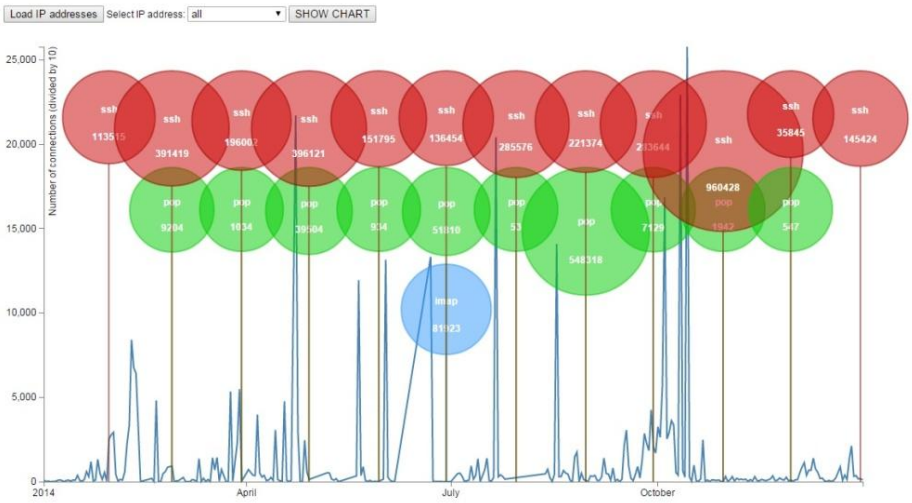
Opäť bolo potrebné získať z URL adresy IP adresu, pre ktorú zobrazujeme graf. Túto IP adresu reprezentuje premenná $arg vo WHERE klauzule dopytu. V prípade, že chceme zobraziť graf súhrnne pre všetky IP adresy, tak si v comboboxe zvolíme „all“, čo sa rovnako pošle v URL adrese PHP skriptu a v IF podmienke sa rozhodne, čo sa má zobraziť.

Na zobrazenie bežca sme využili knižnicu d3.js. Nastavili sme interval, v akom sa majú zobraziť čísla pre jednotlivé mesiace, a rozsah na 12 mesiacov. Práve nastavenú hodnotu bežca posielame ako parameter pre funkciu na vykreslenie grafu nachádzajúcu sa v JavaScripte a ten ju následne posiela PHP skriptu, ktorý túto hodnotu, teda zvolený mesiac, využije v dopyte.

5.4.3 Výsledky

Z vytvorenej vizualizácie je možné vyčítať viacero informácií, pričom sa všetky týkajú použitých, reálnych údajov z honeynetu na Masarykovej univerzite v Brne. Krivka znázorňuje, ako sa počas roka 2014 menil počet pripojení celkovo na honeynet. Na obrázku č. 19 je možné vidieť, že najviac pokusov o pripojenie bolo zaznamenaných v októbri, naopak november bol na počet pokusov o pripojenie k honeynetu najchudobnejší.

V databáze sa nachádzajú aj údaje o tom, na ktorú službu bol pokus o pripojenie zaznamenaný. Práve tieto údaje sú zohľadnené pri vykresľovaní „balónikov“. Vďaka vizualizácii vidíme, že na službu IMAP sa útočilo len v mesiaci jún, naopak na službu SSH boli zaznamenané pokusy o pripojenie počas celého roka. Veľkosť balónikov nám hneď napovie, že najviac ich bolo v októbri. Na službu POP sa útočníci najviac pripájali v auguste.



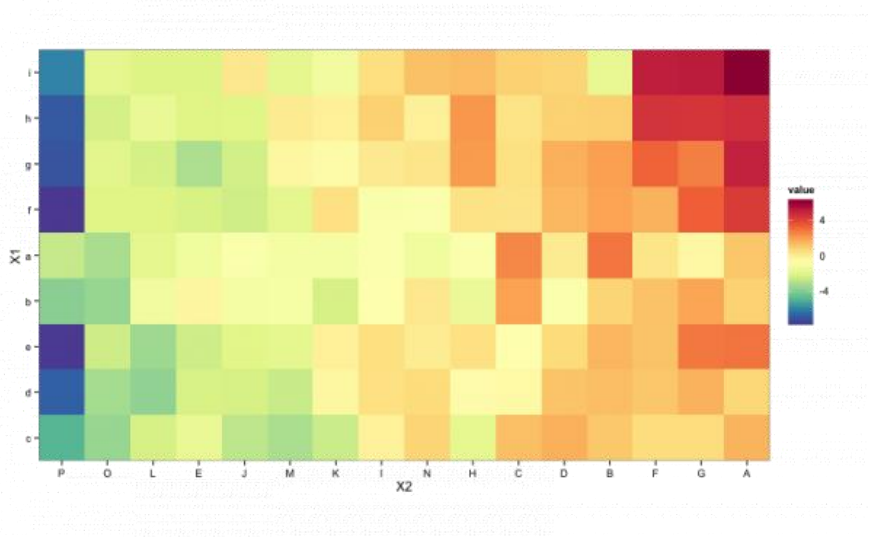
Obrázok 19 Line Plot- vlastná implementácia

5.5 Heatmap

Vo všeobecnosti sa pojem **heatmap** používa pre akékoľvek zobrazenie, ktoré využíva farbu na reprezentáciu kvantitatívnych údajov. Poznáme napríklad heatmapy vo forme máp predpovedajúcich počasie, ktoré používajú farbu na zakódovanie teploty alebo množstva zrážok.

Ak chceme použiť heatmapu na zakódovanie údajov s viacerými premennými, tak sa väčšinou tieto údaje umiestnia do matice, kde stĺpec predstavuje jednu, riadok druhú premennú a prislúchajúca farba zodpovedá hodnote závislej od týchto dvoch

premenných. Na základe legendy je potom možné určiť, akú hodnotu daná farba predstavuje tak, ako to môžeme vidieť na nasledujúcom obrázku č. 20.



Obrázok 20 Heatmap [37]

5.5.1 Využitie v honeynete

Pomocou heatmapy vieme zobraziť rôzne závislosti údajov získaných z honeynetu. Výhodou oproti vizualizácii typu Line Plot je to, že dokážeme zobraziť nie len dve, ale až tri dimenzie, teda pomocou heatmapy vieme zakódovať tri rôzne premenné. V prípade honeynetu vieme heatmapu využiť nasledujúcimi spôsobmi:

- Riadky- dni v týždni, stĺpce- 24 hodín dňa: pri takomto zostavení matice vieme z farby, resp. odtieňa farby a príslušnej legendy vyčítať koľko spojení s honeynetom bolo zaznamenaných v konkrétny deň a hodinu súhrnne napríklad za jeden rok.
- Riadky- mesiace roka, stĺpce- dni v týždni: takto je možné z vizualizácie vyčítať koľko spojení bolo s honeynetom zaznamenaných v daný mesiac a deň.
- Riadky- služby, stĺpce- mesiace roka: keďže honeynet poskytuje rôzne služby, na ktoré sa útočníci pripájajú, je výhodné zobraziť počet spojení s konkrétnou službou v každom mesiaci roka a mať tak prehľad, kedy a na ktorú službu je zaznamenaných najviac pripojení.
- Riadky- GMT zóny, stĺpce- 24 hodín dňa: v tomto prípade je možné z heatmapy vyčítať počet pripojení z konkrétnej GMT zóny v konkrétnu hodinu dňa. Hodiny predstavujú čas na mieste, kde je nasadený

honeynet, teda na Masarykovej univerzite, avšak z GMT zóny vieme ľahko vyčítať, o ktorú hodinu šlo v danej zóne.

Závislosť deň/hodina vieme v našej implementácii zobraziť pre konkrétny, používateľom vybraný filter, teda buď pre jednotlivé krajiny, mestá alebo poskytovateľov internetu, čím rozširujeme vizualizáciu o priestorovo-orientované údaje, ale aj pre konkrétnu IP adresu, GMT zónu a miestnu časovú zónu. Rovnako si vieme vybrať filter pre závislosť deň/mesiac, a to IP adresu, krajinu, mesto alebo poskytovateľa internetu.

Heatmapa je teda vizualizačná technika, ktorú je možné využiť rôznymi spôsobmi. Záleží na tom, aké údaje máme a aké závislosti chceme zobraziť pre rôzne atribúty.

5.5.2 Implementácia

Aj pri tejto implementácii sme využili knižnicu d3.js a na získanie údajov z databázy sme opäť využili PHP skript, v ktorom sa pripájame k databáze a následne pomocou dopytu získavame potrebné údaje. Príklad dopytu, ktorý je použitý pri zobrazovaní toho, ako závisí počet pripojení od dňa a mesiaca pre všetky IP adresy:

```
SELECT weekday(timestamp)+1 as day, month(timestamp) as month,
count(day(timestamp)) as value FROM `dataset_complete` group by
weekday(timestamp)+1, month(timestamp)
```

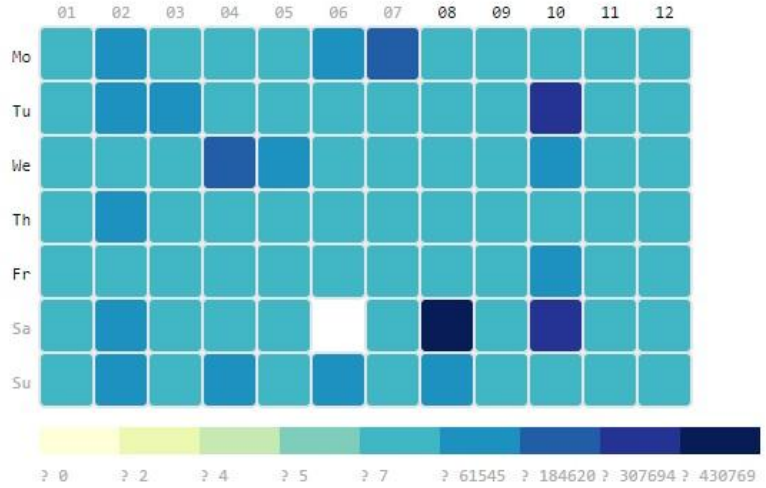
Z výsledku dopytu PHP skript vytvorí pole a následne ho pretransformuje do JSON formátu, ktorý načítame v JavaScripte a získame tak všetky údaje, ktoré následne pomocou d3.js knižnice vizualizujeme vo forme heatmapy. Analogicky sú implementované aj ďalšie možnosti využitia heatmapy popísané vyššie. Výber konkrétnej hodnoty z comboboxu a jej zohľadnenie pri zobrazovaní je riešené rovnako ako pri prvej vizualizačnej metóde LinePlot.

V prípadoch, kde je zvolený filter súvisiaci s priestorovo-orientovanými údajmi, t.j. s krajinou, mestom alebo ISP, využívame aj druhú tabuľku „attacker“, pričom na to, aby sme ku každému záznamu vedeli priradiť časovú pečiatku robíme JOIN medzi tabuľkami „attacker“ a „dataset_complete“ na riadkoch s rovnakými IP adresami.

5.5.3 Výsledky

Z heatmapy, ktorú sme implementovali a ktorá zobrazuje údaje z honeynetu na Masarykovej univerzite, je na prvý pohľad vidno, že najviac pokusov o neautorizované vniknutie do systému bolo v roku 2014 zaznamenaných medzi desiatou a trinástou hodinou vždy v sobotu daného roka.

V prípade závislosti počtu pripojení od dňa a mesiaca bolo najviac pripojení na honeynet v sobotu v mesiaci august. Čo je však zaujímavé a z vizualizácie „bijúce do očí“ , že v mesiaci jún nebol v žiadnu sobotu zaznamenaný žiaden pokus o neautorizované pripojenie, ako máme možnosť vidieť na nasledujúcom obrázku č. 21.



Obrázok 21 Heatmap- vlastná implementácia

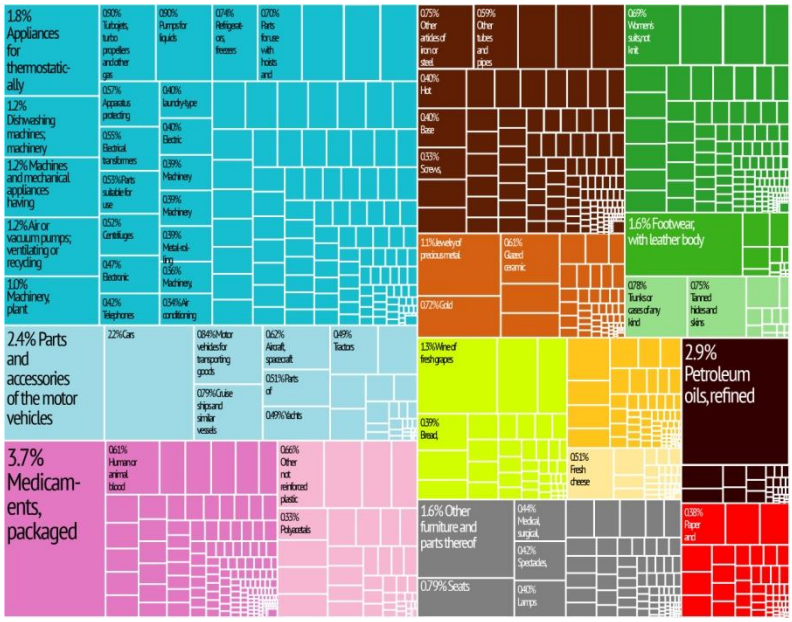
Z vizualizácie heatmapy, ktorá zobrazuje počet pripojení na jednotlivé služby v rámci jednotlivých mesiacov roka, je možné vyčítať, že najmenej pripojení bolo na službu IMAP, a to len v júni. Najviac pripojení bolo zaznamenaných na službu SSH, a to v októbri, čo zodpovedá aj „balóniku“ v predchádzajúcej vizualizačnej metóde.

Zo zobrazenia závislosti GMT/hodiny je vidno, že existujú aj také GMT zóny, z ktorých sa na honeynet v roku 2014 neútočilo vôbec. Podobne vieme vyvodiť rôzne závery, ak sa zameriame na konkrétnu IP adresu, GMT zónu, miestnu časovú zónu, krajinu, mesto či poskytovateľa internetového pripojenia.

5.6 Treemap

Treemap je spôsob vizualizácie multidimenzionálnych, hierarchických údajov, ktorý sa postupne vyvíjal od roku 1990, kedy vznikla potreba vizualizovať stromovú štruktúru priečinkov na disku[17]. Treemapy (obrázok č. 22) poskytujú vizuálnu

reprezentáciu hierarchickej štruktúry v údajoch. Využívajú veľkosť jednotlivých políčok a farbu na zakódovanie špecifických vlastností listových uzlov. Ide predovšetkým o vizualizáciu kvantitatívnych údajov[11]. Vo všeobecnosti je tento spôsob vizualizácie užitočný pri porovnávaní jednotlivých uzlov či celých podstromov, pri vyhľadávaní rôznych vzorov v údajoch, či pri odhaľovaní výnimiek [13].



Obrázok 22 Treemap [38]

5.6.1 Využitie v honeynete

V našej práci využívame treemapu nasledovným spôsobom: K dispozícii máme údaje, v ktorých je pri každom pokuse o pripojenie na honeynet zaznamenaná miestna časová zóna a tiež zodpovedajúce GMT (Greenwich Mean Time). Na základe týchto údajov tak vieme vytvoriť vizualizáciu zobrazujúcu počty pripojení z jednotlivých GMT zón. Ide o hierarchické údaje, keďže do jednej GMT zóny patrí viacero miestnych časových zón. V treemape sa to prejaví tým, že tieto mestá budú mať rovnakú farbu, pričom veľkosť dielika zodpovedá počtu pripojení z danej oblasti (mesta). Keďže máme k dispozícii farby, rozhodli sme sa zafarbiť jednotlivé GMT zóny tým spôsobom, že ak sa tieto zóny geograficky nachádzajú blízko seba, tak majú podobnú farbu, resp. podobný odtieň.

Pomocou bežca je možné postupne zobrazovať treemapu pre jednotlivé mesiace. Vďaka tomu, je možné túto vizualizačnú techniku využiť pre zobrazenie priebehu počtu pripojení na honeynet.

5.6.2 Implementácia

Aj pri tejto vizualizačnej metóde sme využili JavaScriptovú knižnicu d3.js. Na získanie údajov z MySQL databázy bol použitý PHP skript, ktorý vytvorí JSON, ktorý združuje miestne časové zóny prislúchajúce jednotlivým GMT zónam. Dopyty v tomto skripte vyzerajú nasledovne, líšia sa vždy len vo WHERE klauzule, kde sú postupne zadávané jednotlivé GMT zóny.

```
SELECT distinct(time_zone) as name, count(time_zone) as size, gmt as gmt  FROM `dataset_complete` where gmt='+01:00' group by time_zone
```

Výsledky jednotlivých dopytov sú vložené do poľa a následne pretransformované do JSON formátu. Všetky takto vzniknuté JSON-y sme následne v rámci PHP skriptu zreťazili tak, aby každý z nich mal ako rodičovský uzol GMT zónu, ku ktorej patrí.

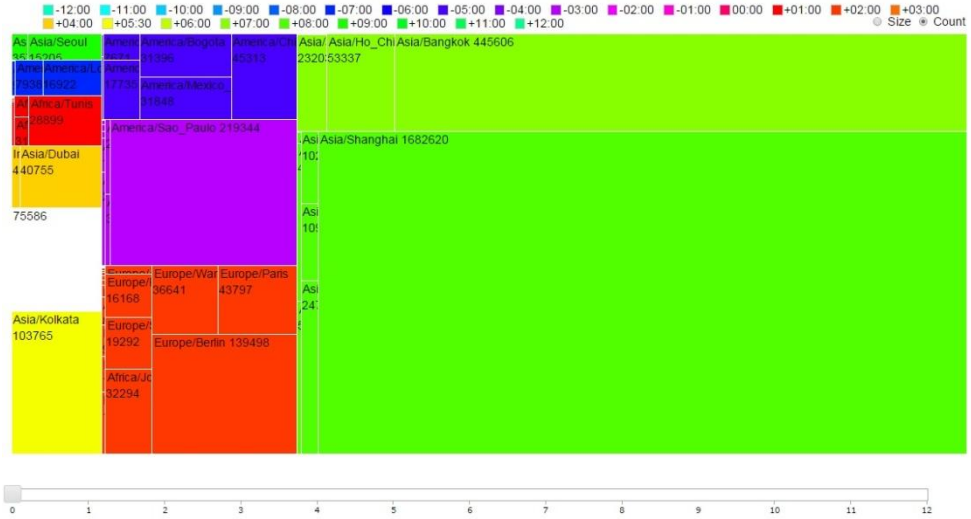
V prípade, že je pomocou bežca zvolený konkrétny mesiac (od 1 po 12), tak v dopyte vo WHERE klauzule sa zohľadní aj táto hodnota a dopyt vyzerá nasledovne:

```
SELECT distinct(time_zone) as name, count(time_zone) as size, gmt as gmt  FROM `dataset_complete` where gmt='+01:00' and month(timestamp)="'.$arg. "'group by time_zone"
```

Na určenie farieb sme využili RGB kruh[16], kde sme 360 stupňov rozdelili medzi 25, resp. 26 GMT zón (posledná farba je priradená mestám, pre ktoré v údajoch nemáme známu GMT zónu) a každej zóne sme tak priradili jednu farbu. V JavaScripte sme následne túto farbu pomocou IF podmienky priradili jednotlivým políčkam.

5.6.3 Výsledky

Z našej implementácie treemapy môžeme vyčítať, že najviac pokusov o útok prichádzalo na honeynet na Masarykovej univerzite z miestneho časového pásma Asia/Shanghai, ako vidíme na obrázku.



Obrázok 23 Treemap- vlastná implementácia

Na tento honeynet sa v roku 2014 útočilo z 10 rôznych GMT zón, pričom v dátach sa vyskytujú aj záznamy s IP adresami, pre ktoré nie je z istých dôvodov uvedená GMT zóna.

6 Záver

Sieťová bezpečnosť dnes predstavuje oblasť, ktorej je potrebné venovať dostatok pozornosti. Nástroj, pomocou ktorého je možné efektívne sledovať kroky útočníka pri pokuse o neautorizované pripojenie do systému, je honeypot, resp. honeynet ako sieť honeypotov. Pomocou týchto systémov vieme získavať množstvo užitočných údajov asociovaných s časom. Otázkou však je, ako tieto časovo-orientované údaje prehľadne zobraziť administrátorovi.

Úvod tejto práce sme venovali teoretickým poznatkom o honeypotoch a honeynetoch. Zaoberali sme sa ich rozdeleniu podľa rôznych kritérií, ich výhodám, ale aj nevýhodám. Ťažiskom tejto práce bola však vizualizácia časovo-orientovaných údajov. Najprv sme urobili analýzu rôznych vizualizačných metód aplikovateľných na rôzne typy údajov a vybrali z nich tie, ktoré sú vhodné na časovo-orientované údaje získané z honeynetu. Implementovali sme tri z nich: LinePlot, ktorého výhodou je, že klasický graf s dvoma osami a krivkou popisujúcou počet pripojení na honeynet v čase, bolo možné obohatiť o zobrazenie štatistiky vždy na konci týždňa. Vďaka tejto štatistike má používateľ prehľad o tom, na aké služby bolo zaznamenaných koľko pripojení za daný týždeň. Ďalšia vizualizačná technika, heatmap, priniesla tú výhodu, že v dvojrozmernom priestore dokážeme zobraziť až 3 atribúty naraz, napríklad deň, mesiac a počet pripojení na honeynet v daný deň mesiaca súhrnne za celý rok. Poslednú metódu, treemap, sme využili na zobrazenie počtu pripojení z jednotlivých časových zón, resp. GMT zón. Farby sme v tomto prípade využili tak, že mestá nachádzajúce sa v rovnakej zóne majú rovnakú farbu a zóny, ktoré sú geograficky blízko seba majú podobnú farbu, resp. podobný odtieň. Množstvo pripojení na honeynet je v tomto prípade vyjadrený veľkosťou políčka.

Implementácie jednotlivých vizualizačných metód je možné priamo nasadiť v systémoch na analýzu údajov z honeypotov alebo honeynetov.

7 Zoznam použitej literatúry

[1] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, Christian Tominski: Visualization of Time-Oriented Data, 2011.

[2] Wolfgang Aigner et al. Visual Methods for Analyzing Time-Oriented Data[online]. Dostupné na internete: http://www.informatik.uni-rostock.de/~schumann/papers/2008+/TVCGTimeViz.pdf

[3] Bertin, J. (1983). Semiology of Graphics: Diagrams, Networks, Maps. University of Wisconsin Press, Madison, WI, USA. translated by William J. Berg.

[4] J. Blasco. An approach to malware collection log visualization. Journal, 2005.

[5] Card, S. K., Suh, B., Pendleton, B. A., Heer, J., and Bodnar, J. W. (2006). Time Tree: Exploring Time Changing Hierarchies. In Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST), strany 3-10, Los Alamitos, CA, USA. IEEE Computer Society.

[6] Chi, E. H. (2000). A Taxonomy of Visualization Techniques Using the Data State Reference Model. In Proceedings of the IEEE Symposium on Information Visualization (InfoVis), pages 69–76, Washington, DC, USA. IEEE Computer Society.

[7] Erbacher, R. F., Walker, K. L., and Frincke, D. A. (2002). Intrusion and Misuse Detection in Large-Scale Systems. IEEE Computer Graphics and Applications, 22(1):38-48.

[8] F. Fischer, F. Mansmann, D.A. Keim, S. Pietzko, and M. Waldvogel. Large-scale network monitoring for visual analysis of attacks. Lecture Notes in Computer Science,5210:111, 2008.

[9] J. R. Goodall and M. Sowul. Viassist: Visual Analytics foe cyber defense. In Technologies for Homeland Security, 2009. HST'09. IEEE Conference on, strany 143-150. IEEE, 2009.

[10] Tomasz Grudziecki et al. Proactive Detection of Security Incidents Honeypots 2012-11-20

[11] Jay Jacobs, Bob Rudis (2014). Data-Driven Security: Analysis, Visualization and Dashboards

[12] JOSHI, R.C. – SARDANA A. 2011. Honeypots: A New Paradigm to Information Security. Science Publishers, 2011.

[13] Marty, R. (2009). Applied security visualization (pp. 21-65). Upper Saddle River: Addison-Wesley.

[14] Mennis, J. L., Peuquet, D., and Qian, L. (2000). A Conceptual Framework for Incorporating Cognitive Principles into Geographical Database Representation. International Journal of Geographical Information Science, 14(6):501–520.

[15] Robertson, P. K. (1991). A Methodology for Choosing Data Representations. IEEE Computer Graphics and Applications, 11(3):56–67.

[16] Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In Proceedings of the IEEE Symposium on Visual Languages, pages 336–343, Los Alamitos, CA, USA. IEEE Computer Society.

[17] Ben Shneiderman (1998). Treemaps for space-constrained visualization of hierarchies[online]. Dostupné na internete: http://www.cs.umd.edu/hcil/treemap-history/

[18] Spitzner, L. Honeypots: Tracking Hackers, Addison Wesley, 2002, pp. 1-430.

[19] V. Visoottiveseth, U. Jaralrungroj, E. Phoomrungraungsuk, and P. Kultanon. Distributed honeypot log management and visualization of attacker geographical distribution. In Computer Science and Software Engineering (JCSSE), 2011 Eight International Joint Conference on, strany 23-28. IEEE, 2011.

[20] Wong, P. C. and Bergeron, R. (1997). 30 Years of Multidimensional Multivariate Visualization. In Nielson, G. M., Hagen, H., and Muller, H., editors, ¨ Scientific Visualization, pages 40–62. IEEE Computer Society, Los Alamitos, CA, USA.

[21] Dostupné na internete: http://ieg.ifs.tuwien.ac.at/~aigner/presentations/TimeViz06.pdf

[22] Dostupné na internete: http://www.rapidtables.com/web/color/color-wheel.htm

[23] Dostupné na internete: Kippo-graph visualization. http://bruteforce.gr/kippo-graph.

[24] Dostupné na internete: http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1533&context=ecuworks

[25]Dostupné na internete: Surfcert intrusion detection system. http://ids.surfnet.nl/.

[26] Dostupné na internete: http://www.honeynet.org/node/960

[27] Dostupné na internete: https://www.pravail.com/#about-us

[28] Dostupné na internete: https://gephi.github.io/features/

[29] Dostupné na internete: http://sourceforge.net/projects/xmdvtool/

[30] Dostupné na internete: http://www.cg.cs.tu-bs.de/publications/Lipski09wscg/

[31] Dostupné na internete: https://polaris.codeplex.com/

[32] Dostupné na internete: http://www.honeyd.org/

[33] Dostupné na internete: http://www.specter.com/

[34] Dostupné na internete: https://github.com/desaster/kippo

[35] Dostupné na internete: http://www.honeynet.org/project/sebek/

[36] Dostupné na internete: http://www.informatik.uni-rostock.de/~ct/software/VisAxes/VisAxesNET.html

[37] Dostupné na internete: http://www.xeqtit.com/uploads/default/files/HeatMap.png

[38] Dostupné na internete: http://upload.wikimedia.org/wikipedia/commons/9/9d/Italy_Export_Treemap.jpg

[39] Dostupné na internete: http://ip-api.com/

Prílohy

Príloha A: CD médium – bakalárska práca v elektronickej podobe, prílohy v elektronickej podobe, kódy implementované v práci.

Príloha B: Skripty na získanie údajov z databázy

Príloha C: Skript na naplnenie „comboboxu“ hodnotami

Príloha B: Skripty na získanie údajov z databázy

PHP skript na získanie údajov z databázy pre vykreslenie krivky LinePlot-u

```
<?php
$username = "time_oriented";
$password = "Heslo2015";
$host = "localhost";
$database="time_oriented";

$server = mysql_connect($host, $username, $password);
$connection = mysql_select_db($database, $server);
//získanie IP adresy
if (isset($_GET["w1"])) {
    $arg = $_GET["w1"] ;
    }
    ;
//získanie mesiaca
if (isset($_GET["w2"])) {
    $arg2 = $_GET["w2"] ;
    }
    ;


if (isset($_GET["w1"]) && $_GET["w1"]!=="all" && isset($_GET["w2"])) {

$myquery = "select * from (select date(timestamp) as date,
count(timestamp) as pct50 from `dataset_complete`where ip='".$arg.
"'and month(timestamp)='".$arg2. "' group by date(timestamp)) t1 where
t1.pct50!=(select max(pct50) as max from (select date(timestamp) as
date,  count(timestamp) as pct50 from `dataset_complete` where
ip='".$arg. "' and month(timestamp)='".$arg2. "' group by
date(timestamp)) t1) and t1.pct50!=(select min(pct50) as max from
(select date(timestamp) as date,  count(timestamp) as pct50 from
`dataset_complete` where ip='".$arg. "' and month(timestamp)='".$arg2.
"'group by date(timestamp)) t1)
 ";

if (!$myquery) {
   echo "no data";
   }
}
if (isset($_GET["w1"]) && $_GET["w1"]=="all" && isset($_GET["w2"])) {
//vynecha max a min hodnotu
$myquery = "select * from (select date(timestamp) as date,
count(timestamp) as pct50 from `dataset_complete` where
month(timestamp)= '".$arg2. "'group by date(timestamp)) t1 where
t1.pct50!=(select max(pct50) as max from (select date(timestamp) as
date,  count(timestamp) as pct50 from `dataset_complete`where
month(timestamp)= '".$arg2. "' group by date(timestamp)) t1) and
t1.pct50!=(select min(pct50) as max from (select date(timestamp) as
date,  count(timestamp) as pct50 from `dataset_complete`where
month(timestamp)= '".$arg2. "' group by date(timestamp)) t1);
 ";

}

if (isset($_GET["w1"]) && $_GET["w1"]!=="all" && !isset($_GET["w2"]))
{
$myquery = "select * from (select date(timestamp) as date,
count(timestamp) as pct50 from `dataset_complete`where ip='".$arg. "'
group by date(timestamp)) t1 where t1.pct50!=(select max(pct50) as max
from (select date(timestamp) as date,  count(timestamp) as pct50 from
`dataset_complete` where ip='".$arg. "' group by date(timestamp)) t1)
and t1.pct50!=(select min(pct50) as max from (select date(timestamp)
as date,  count(timestamp) as pct50 from `dataset_complete` where
ip='".$arg. "'group by date(timestamp)) t1)
```

```
 ";

}

if (isset($_GET["w1"]) && $_GET["w1"]=="all" && !isset($_GET["w2"])) {
$myquery = "select * from (select date(timestamp) as date,
count(timestamp) as pct50 from `dataset_complete` group by
date(timestamp)) t1 where t1.pct50!=(select max(pct50) as max from
(select date(timestamp) as date,  count(timestamp) as pct50 from
`dataset_complete` group by date(timestamp)) t1) and t1.pct50!=(select
min(pct50) as max from (select date(timestamp) as date,
count(timestamp) as pct50 from `dataset_complete` group by
date(timestamp)) t1);
 ";
}

    $query = mysql_query($myquery);

    if ( ! $query ) {
        echo mysql_error();
        die;
    }

    $data = array();

    for ($x = 0; $x < mysql_num_rows($query); $x++) {
        $data[] = mysql_fetch_assoc($query);
    }

echo json_encode($data);

    mysql_close($server);

?>
```

PHP skript na získanie údajov pre vykreslenie heatmapy- závislosť Dni/Hodiny pre filtre IP adresa, GMT zóna a miestna časová zóna

```
<?php
$username = "time_oriented";
$password = "Heslo2015";
$host = "localhost";
$database="time_oriented";

$server = mysql_connect($host, $username, $password);
$connection = mysql_select_db($database, $server);
if (isset($_GET["w1"])) {
    $arg = $_GET["w1"] ;
    }
    ;

if (isset($_GET["w2"])) {
    $arg2 = $_GET["w2"] ;

    }
    ;
if (isset($_GET["w3"])) {
    $arg3 = $_GET["w3"] ;

    }
    ;
if (isset($_GET["w1"]) &&  $_GET["w1"]!=="all" ) {
$myquery = "SELECT weekday(timestamp)+1 as day, hour(timestamp)+1 as
hour, count(day(timestamp)) as value FROM `dataset_complete` where
ip='".$arg. "' group by weekday(timestamp)+1,hour(timestamp)+1

";
    }
```

```
if (isset($_GET["w1"]) &&  $_GET["w1"]=="all" ) {
$myquery = "SELECT weekday(timestamp)+1 as day, hour(timestamp)+1 as
hour, count(day(timestamp)) as value FROM `dataset_complete` group by
weekday(timestamp)+1,hour(timestamp)+1

";
 }
if (isset($_GET["w2"]) &&  $_GET["w2"]!=="all") {
  $myquery = "SELECT weekday(timestamp)+1 as day, hour(timestamp)+1 as
hour, count(day(timestamp)) as value FROM `dataset_complete` where
gmt='".$arg2. "' group by weekday(timestamp)+1,hour(timestamp)+1

";
 }
if (isset($_GET["w2"]) &&  $_GET["w2"]=="all" ) {

$myquery = "SELECT weekday(timestamp)+1 as day, hour(timestamp)+1 as
hour, count(day(timestamp)) as value FROM `dataset_complete` group by
weekday(timestamp)+1,hour(timestamp)+1

";
 }

if (isset($_GET["w3"]) &&  $_GET["w3"]!=="all") {
$myquery = "SELECT weekday(timestamp)+1 as day, hour(timestamp)+1 as
hour, count(day(timestamp)) as value FROM `dataset_complete` where
time_zone='".$arg3. "' group by weekday(timestamp)+1,hour(timestamp)+1

";
 }
if (isset($_GET["w3"]) &&  $_GET["w3"]=="all" ) {

$myquery = "SELECT weekday(timestamp)+1 as day, hour(timestamp)+1 as
hour, count(day(timestamp)) as value FROM `dataset_complete` group by
weekday(timestamp)+1,hour(timestamp)+1

";
 }

    $query = mysql_query($myquery);

    if ( ! $query ) {
        echo mysql_error();
        die;
    }

    $data = array();

    for ($x = 0; $x < mysql_num_rows($query); $x++) {
        $data[] = mysql_fetch_assoc($query);
    }

echo json_encode($data);

    mysql_close($server);

?>
```

Príloha C: Skript na naplnenie „comboboxu“ hodnotami

Funkcia, ktorá zabezpečí naplnenie comboboxu IP adresami, ktoré sa nachádzajú v databáze

```
function readIP() {

    var address=[];
    $.getJSON( "readIP.php", function( data ) {
    console.log('loaded');

    $.each(data.children, function (index, child) {
        address.push(child.ip);
    });
    var opt = document.createElement("option");
        document.getElementById("mySelect").options.add(opt);
        opt.text="all";
        opt.value="all";
    for(i = 0; i < address.length; i++) {
        var opt = document.createElement("option");

        document.getElementById("mySelect").options.add(opt);

        opt.text = address[i].toString();
        opt.value = address[i].toString();
        }

    console.log(address);
});



    }
```