This file has been cleaned of potential threats.

If you confirm that the file is coming from a trusted source, you can send the following SHA-256 hash value to your admin for the original file.

b3c037c00892772fd8002a11469d5e63c3547b07723b24721322ec45bc0a2d94

To view the reconstructed contents, please SCROLL DOWN to next page.

دانشکده مهندسی

گروه مهندسی کامپیوتر

پایان‌نامه کارشناسی ارشد

استخراج آنتولوژی به روش داده‌کاوی
به منظور استفاده در سیستم تشخیص نفوذ همکارانه

تهیه و تنظیم:

مهدی هاشمی شهرکی

استاد راهنما:

دکتر محسن کاهانی

استاد مشاور:

پرفسور محمود نقیب‌زاده

زمستان ۸۸

بسم الله الرحمن الرحيم

تقدیم به

روح بزرگ پدرم

و

صبر بی پایان مادرم

تقدیر و تشکر

در این‌جا بر خود لازم می‌دانم از تلاش اساتید محترم و با تجربه گروه کامپیوتر دانشگاه فردوسی مشهد به ویژه جناب آقای دکتر کاهانی و جناب آقای پرفسور نقیب‌زاده که همواره راهنمای راهم بودند تشکر کنم. در پایان نیز از خانواده‌ام و کلیه دوستانی که مرا در انجام این تحقیق یاری رساندند تشکر و قدردانی می‌کنم.

چکیده

امروزه با گسترش شبکه‌های کامپیوتری، بحث امنیت شبکه بیش از گذشته مورد توجه پژوهشگران قرار گرفته است. در این راستا تشخیص نفوذ به‌عنوان یکی از اجزای اصلی برقراری امنیت در شبکه‌های کامپیوتری شناخته می‌شود که هدف اصلی آن کنترل ترافیک شبکه و تحلیل رفتارهای کاربران می‌باشد. به‌طور کلی در اغلب کارهای انجام گرفته در این حوزه از یک تاکسونومی جهت نمایش ویژگی حملات استفاده شده است. بکارگیری تاکسونومی مشکلات و محدودیت‌های فراوانی در سیستم تشخیص نفوذ ایجاد می‌کند که برای جلوگیری از بروز آنها می‌توان از یک آنتولوژی جهت طبقه‌بندی و بیان ویژگی حملات استفاده نمود. آنتولوژی علاوه بر دارا بودن ویژگی‌های تاکسونومی مزایای عمده دیگری نیز دارد و با استفاده از آن می‌توان مدل داده-ای تشخیص نفوذ را از منطق سیستم کشف نفوذ تفکیک نمود.

از این‌رو هدف اصلی این تحقیق بر روی استخراج آنتولوژی حملات در حوزه‌ی تشخیص نفوذ شبکه‌های کامپیوتری بنا شده است. بدین منظور یک معماری توزیع شده مبتنی بر عامل طراحی شده است، و از تکنیک‌های داده‌کاوی متفاوتی مانند الگوریتم‌های طبقه‌بندی *RIPPER* و خوشه‌بندی *HotSpot* بر روی مجموعه‌ی داده‌ای *NSL-KDD* استفاده می‌شود. با بهره‌گیری از این الگوریتم‌ها می‌توان قوانین لازم جهت مشخص نمودن ویژگی کلاس‌های مختلف آنتولوژی حملات را تولید نمود. آنتولوژی بدست آمده به عنوان یک طبقه‌بند در سیستم تشخیص نفوذ توزیع شده به‌کار گرفته می‌شود. بدین منظور آنتولوژی حملات در قالب یک فایل *OWL* در اختیار عامل مرکزی قرار می‌گیرد. وظیفه عامل مرکزی این است که گزارش عامل‌های ایستا را دریافت کرده و با بهره‌گیری از زبان پرس‌وجوی *SPARQL* و تکنیک‌های تشابه معنایی، کلاس مربوط به نمونه‌ی دریافتی را در آنتولوژی حملات مشخص کند. ارزیابی سیستم پیشنهادی بر روی مجموعه داده‌ای *NSL-KDD* حاکی از قدرت آن در تشخیص نفوذ شبکه‌های کامپیوتری می‌باشد. به‌طوری که با بهره‌گیری از این سیستم می‌توان به نرخ تشخیص ۹۹/۲٪ همراه با نرخ هشدار غلط ۰/۲ دست یافت.

کلمات کلیدی: تشخیص نفوذ در شبکه‌های کامپیوتری، داده‌کاوی، آنتولوژی حملات کامپیوتری، سیستم‌های تشخیص نفوذ توزیع شده، تشابه معنایی، *RIPPER* و *HotSpot*

فهرست مطالب

فصل اول: «مقدمه»..... ۱

۱- ۱-تاریخچه‌ی تشخیص نفوذ در شبکه‌های کامپیوتری..... ۲

۱- ۲-استفاده از آنتولوژی به‌جای تاکسونومی در تشخیص نفوذ..... ۳

۱- ۳-تحقیق انجام شده در این پایان نامه..... ۵

۱- ۴-نمای کلی پایان نامه..... ۶

فصل دوم: «پیش زمینه و بررسی کارهای مشابه»..... ۸

۲- ۱-مقدمه..... ۹

۲- ۲-تشخیص نفوذ..... ۹

۲- ۳-طبقه‌بندی سیستم‌های تشخیص نفوذ..... ۱۱

۲- ۳- ۱-معماری سیستم‌های تشخیص نفوذ..... ۱۲

۲- ۳- ۲-رویکرد تشخیص نفوذ..... ۱۳

۲- ۳- ۲- ۱-تشخیص سوءاستفاده..... ۱۳

۲- ۳- ۲- ۲-تشخیص رفتار غیرعادی..... ۱۴

۲- ۳- ۲- ۳-تشخیص ترکیبی..... ۱۴

۲- ۳- ۳-منبع تشخیص نفوذ..... ۱۵

۲- ۳- ۳- ۱-تشخیص نفوذ بر اساس مدل میزبان..... ۱۵

۲- ۳- ۳- ۲-تشخیص نفوذ بر اساس ترافیک شبکه..... ۱۶

۲- ۳- ۴-ساختار سیستم تشخیص نفوذ..... ۱۷

۲- ۳- ۴- ۱-عامل‌ها و سیستم‌های تشخیص نفوذ توزیع شده..... ۱۷

۲- ۳- ۵-نوع داده..... ۱۸

۲- ۳- ۶-مکانیزم پاسخ..... ۱۸

۲- ۳- ۷-پویایی سیستم..... ۱۹

۲- ۴-معیارهای ارزیابی سیستم تشخیص نفوذ..... ۱۹

۲- ۵-خلاصه..... ۲۳

فصل سوم: «تکنیک‌های داده‌کاوی در تشخیص نفوذ»..... ۲۴

۳- ۱-مقدمه..... ۲۵

۳- ۲-داده‌کاوی..... ۲۵

۳- ۳-داده‌کاوی در سیستم‌های تشخیص نفوذ..... ۲۷

۳- ۳- ۱-انتخاب خصیصه‌ها..... ۲۷

۳- ۳- ۲-یادگیری ماشین..... ۲۸

۳- ۳- ۲- ۱-تکنیک‌های طبقه‌بندی..... ۲۸

۳- ۳- ۲- ۲-تکنیک‌های خوشه‌بندی..... ۳۵

۳- ۳- ۳-تکنیک‌های آماری..... ۳۷

۳- ۴-تکنیک داده‌کاوی استفاده شده در این تحقیق..... ۳۸

۳- ۴- ۱-الگوریتم *REP*..... ۳۸

۳- ۴- ۲-الگوریتم *IREP*..... ۳۹

۳- ۴- ۳-الگوریتم *RIPPER*..... ۴۰

۳- ۵-خلاصه..... ۴۳

فصل چهارم: «وب معنایی و کاربرد آن در تشخیص نفوذ» ۴۵
۴- ۱-مقدمه ۴۶
۴- ۲-مروری بر وب معنایی ۴۶
۴- ۲- ۱-آنتولوژی ۴۷
۴- ۳-وب معنایی در امنیت اطلاعات ۴۸
۴- ۳- ۱-وب معنایی در سیستم‌های تشخیص نفوذ ۴۹
۴- ۳- ۲-مرورتحقیقات انجام شده ۵۰
۴- ۴-خلاصه ۵۷
فصل پنجم: « طراحی آنتولوژی حملات کامپیوتری» ۵۸
۵- ۱-مقدمه ۵۹
۵- ۲-محیط طراحی آنتولوژی ۵۹
۵- ۳-طراحی و ایجاد آنتولوژی ۶۰
۵- ۳- ۱-تعریف کلاس‌ها در آنتولوژی ۶۱
۵- ۳- ۲-مرتب کردن سلسله مراتب کلاس‌ها در آنتولوژی ۶۲
۵- ۳- ۳-تعریف ویژگی‌ها و خصایص و معین کردن مقادیر مجاز برای آنها ۶۵
۵- ۳- ۳- ۱-مجموعه داده‌های *KDD'99* ۶۶
۵- ۳- ۳- ۲-مجموعه داده‌های آزمایشی *NSL-KDD* ۶۸
۵- ۳- ۳- ۳-یادگیری از داده‌های *NSL-KDD* با استفاده از قوانین *RIPPER* ۶۹
۵- ۳- ۴-مقداردهی به خصایص برای نمونه‌های تعریف شده در آنتولوژی ۷۳
۵- ۳- ۵-بررسی صحت و درستی آنتولوژی طراحی شده ۷۶
۵- ۴-خلاصه ۷۷
فصل ششم: « پیاده‌سازی و بررسی نتایج» ۷۸
۶- ۱-مقدمه ۷۹
۶- ۲-تکنولوژی عامل‌ها ۷۹
۶- ۳-طراحی محیط مبتنی بر عامل ۸۰
۶- ۳- ۱-بستر پیاده سازی عامل‌ها ۸۱
۶- ۳- ۲-معرفی سیستم پیشنهادی ۸۱
۶- ۳- ۲- ۱-معماری *DIDMO* ۸۱
۶- ۳- ۲- ۲-تعاملات عامل‌ها و تشخیص نفوذ ۸۳
۶- ۳- ۲- ۳-جزئیات بیشتر پیاده‌سازی ۸۵
۶- ۳- ۲- ۴-مدل دادهای موجود در عامل‌های ایستا ۸۶
۶- ۳- ۲- ۵-نحوه‌ی عملکرد عامل مرکزی ۸۸
۶- ۴-آزمایش‌ها و بررسی نتایج ۹۳
۶- ۴- ۱-بررسی نتایج و ارزیابی سیستم ۹۴
۶- ۴- ۲-مقایسه‌ی سیستم پیشنهادی با سیستم‌های مشابه ۹۷
۶- ۵-خلاصه ۹۹
فصل هفتم: « نتیجه گیری و کارهای آتی» ۱۰۱
۷- ۱-اهداف پایان‌نامه ۱۰۲
۷- ۲-تحقیقات بیشتر ۱۰۳
منابع ۱۰۵

ضمائم ۱۱۲
ضمیمه الف: مشخصات مجموعه دادهای *NSL-KDD* ..... *I*
ضمیمه ب: قوانین بدست آمده با استفاده از روش طبقه‌بندی *RIPPER* ..... *III*

فهرست جداول

جدول ۲- ۱: ماتریس هزینه‌ی استفاده‌شده در مسابقه یادگیری طبقه‌بندی کننده‌های *KDD'99* ۲۲
جدول ۵- ۱ : آمار رکوردهای زائد در مجموعه آموزش *KDD* ۶۹
جدول ۵- ۲: آمار رکوردهای زائد در مجموعه تست *KDD* ۶۹
جدول ۶- ۱ : لیست خصایص قابل اصلاح ۸۷
جدول ۶- ۲ : مدل داده‌ای موجود در عامل‌های ایستا ۸۸
جدول ۶- ۳: قرابت موجود بین پروتکلهای *ICMP*، *TCP* و *UDP* ۹۳
جدول ۶- ۴: نتایج طبقه بندی مجموعه داده‌های تست ۹۴
جدول ۶- ۵ : ماتریس بر هم ریختگی مربوط به طبقه‌بندی ۲۳ کلاسه ۹۶
جدول ۶- ۶: ماتریس بر هم ریختگی پنج کلاسه ۹۶
جدول ۶- ۷: تست سیستم *DIDMO* با استفاده از مجموعه داده‌ای *NSL-KDD* ۹۶
جدول ۶- ۸: نرخ طبقه بندی، *DTR*، *FA* و *CPE* برای طبقه‌بندی مجموعه داده‌ای *NSL-KDD* ۹۸
جدول ۶- ۹: ماتریس برهم ریختگی برای مقایسه‌ی روش *DIDMO* با روش *SWIDS* ۹۹

فهرست شکل‌ها

شکل ۲- ۱: یک سیستم تشخیص نفوذ ساده ۱۱
شکل ۲- ۲ : طبقه‌بندی سیستم‌های تشخیص نفوذ ۱۲
شکل ۲- ۳: وضعیت‌های ممکن در قبال واکنش‌های یک سیستم تشخیص نفوذ ۲۰
شکل ۲- ۴: منحنی *ROC* ۲۱
شکل ۳- ۱ : فرایند تبدیل داده‌های خام به دانش مفید ۲۶
شکل ۳- ۲: سیستم استنتاج فازی ممدانی ۳۲
شکل ۳- ۳: بهترین ابرصفحه برای جداسازی داده‌ها ۳۵
شکل ۳- ۴: انواع خوشه‌بندی ۳۶
شکل ۳- ۵: الگوریتم *IREP* ۳۹
شکل ۳- ۶: الگوریتم *RIPPER* ۴۲
شکل ۳- ۷ : مقایسه‌ی میزان خطای الگوریتم‌های *RIPPERK* و *C4.5* ۴۲
شکل ۳- ۸ : مقایسه‌ی سرعت الگوریتم‌های *RIPPERK* و *C4.5* ۴۳
شکل ۴- ۱: لایه‌های تشکیل دهنده‌ی وب معنایی ۴۷
شکل ۴- ۳: آنتولوژی *TARGET_ CENTRIC* ۵۱
شکل ۴- ۴: آنتولوژی *ATTACKER-CENTRIC* ۵۳
شکل ۴- ۵: آنتولوژی اثر حمله ۵۴
شکل ۴- ۶: قسمتی از آنتولوژی مورد استفاده در *[YAN04]* ۵۵
شکل ۵- ۱: شمایی از کلاس حملات *R2L* و زیرکلاس‌های آن ۶۳
شکل ۵- ۲: اسکلت اصلی آنتولوژی "حملات کامپیوتری" ۶۴
شکل ۵- ۳ : ساختار کلاس‌های *VALUEPARTITION* ۶۴
شکل ۵- ۴: تنظیمات مربوطه در *RAPIDMINER* ۷۰
شکل ۵- ۵: چند نمونه از قوانین *RIPPER* مستخرج از داده‌کاوی ۷۱
شکل ۵- ۶ : تعریف مقادیر ابتدایی و انتهایی برای هر بازه ۷۲
شکل ۵- ۷ : مقادیر مجاز برای کلاس‌های *VALUEPARTITION* ۷۳
شکل ۵- ۸ : تعریف ویژگی شئی *HAS_SRC_BYTES* ۷۵
شکل ۶- ۱ : معماری *DIDMO* ۸۲
شکل ۶- ۲: شباهت جزئی خصیصه‌های عددی ۹۰

فصل اول

«مقدمه»

هم‌زمان با رشد شبکه‌های کامپیوتری حملات و نفوذ به آن‌ها نیزافزایش چشمگیری داشته است. ازاین‌رو امنیت شبکه[۱] یکی از مباحثی است که امروزه بسیار مورد توجه محققین و پژوهش‌گران قرار گرفته است. امنیت شبکه شامل حوزه تحقیقاتی گسترده‌ای است که محافظت از ساختار شبکه‌های کامپیوتری و کلیه سرویس‌های مربوط به آن‌را شامل می‌شود. به‌طورکلی امنیت در شبکه‌های کامپیوتری شامل چهار جزء اصلی به شرح زیر می‌باشد: *[Wan09]*.

- امنیت ویژه زیرساخت‌ها:[۲] از قبیل دیواره‌های آتش سخت‌افزاری و نرم‌افزاری و همچنین روش-های فیزیکی برقراری امنیت.
- سیاست‌های امنیتی:[۳] شامل پروتکل‌های امنیتی، اعتبارسنجی کاربران، کنترل دسترسی‌ها و نگهداری از اطلاعات محرمانه.
- تشخیص برنامه‌های مخرب:[۴] شامل برنامه‌هایی برای از بین بردن ویروس‌ها و کرم‌های رایانه‌ای.
- تشخیص نفوذ:[۵] تشخیص موارد نفوذ به شبکه با بررسی مداوم ترافیک شبکه و نظارت بر رفتار کاربران.

۱-۱- تاریخچه‌ی تشخیص نفوذ در شبکه‌های کامپیوتری

James P. Anderson برای اولین بار مسئله‌ی توجه به مشکلات امنیتی کامپیوتر را مطرح کرد. وی در سال ۱۹۷۲ طرحی برای رویارویی با مسائل امنیتی سیستم‌های کامپیوتری نیروی هوایی آمریکا ارائه داد. امروزه پس از گذشت سه دهه از آن زمان و با وجود پیشرفت‌های زیادی که در این زمینه حاصل شده، تأمین امنیت شبکه‌های کامپیوتری همچنان مورد توجه محققین و پژوهش‌گران است *[Wan09]*.

آقای *Anderson* در سال ۱۹۸۰ و پس از اتمام پروژه تأمین امنیت سیستم‌های کامپیوتری نیروی هوایی آمریکا گزارشی باعنوان "نظارت و بررسی تهدیدهای امنیت شبکه" ارائه داد. وی در این گزارش برای اولین بار

---

[۱] *Network Security*

[۲] *Security-Specific Infrastructures*

[۳] *Security Polices*

[۴] *Detection of Malicious Programs*

[۵] *Intrusion Detection*

به اهمیت داده‌هایی که به‌صورت روزانه از کامپیوترهای موجود در شبکه استخراج می‌شوند، برای تشخیص تهدیدات امنیتی پی‌برد. به هر حال تا آن زمان اهمیت چنین داده‌هایی درک نشده بود و همه تلاش‌ها بر روی جلوگیری از دسترسی به داده‌های حساس از طریق منابع غیرمجاز خلاصه می‌شد.

در دهه ۱۹۹۰ امنیت شبکه در حوزه تشخیص نفوذ در شبکه‌های کامپیوتری وارد مرحله جدیدی شد. در این دوران افزایش حملات به شبکه‌های کامپیوتری موجب شد بسیاری از محققان در دانشگاه‌های مختلف بر روی این موضوع متمرکز شوند. امروزه با توجه به گسترش فراوان شبکه‌های کامپیوتری همچنین افزایش پهنای باند شبکه‌ها، تشخیص حملات بسیار دشوار شده است. ازاین‌رو طراحی روش‌هایی که بتوانند با دقت بالا و در زمانی معقول موارد نفوذ به شبکه را تشخیص دهند از اهمیت ویژه‌ای برخوردار است.

۱– ۲– استفاده از آنتولوژی به‌جای تاکسونومی در تشخیص نفوذ

مهمترین مؤلفه یک سیستم تشخیص نفوذ تاکسونومی می‌باشد. این تاکسونومی برای طبقه‌بندی و بیان ویژگی‌های حملات و نفوذها مورد استفاده قرار می‌گیرد. برای تشریح نمونه‌های این تاکسونومی احتیاج به یک زبان می‌باشد که این زبان مهمترین نقش را در کارایی سیستم تشخیص نفوذ ایفا می‌کند. زیرا اطلاعات مربوط به یک حمله یا نفوذ بایستی در یک محیط توزیع شده بطور واضح هدایت شود و عمل مناسب بر روی آنها انجام گیرد. تاکنون چندین تاکسونومی توسط محققان ارائه شده که برخی از آنها شامل یک زبان توصیفی می‌باشند ولی اغلب فاقد چنین زبانی هستند*[Han01][Ken99][Nig01]*. اگرچه چندین زبان حمله ارائه شده است ولی بیشتر آنها برای انواع تاکسونومی‌ها مناسب نمی‌باشد و تنها مخصوص تاکسونومی مربوطه بوده و به صورت محلی عمل می‌نمایند. می‌توان مشکلات ذاتی این روش را در سه گروه دسته بندی کرد:

- برای انجام عملیات بر روی نمونه‌های مدل داده‌ای توصیف شده در یک تاکسونومی خاص، مدل داده‌ای بایستی در یک سیستم نرم‌افزاری کدگذاری شود در نتیجه هر تغییر یا به روز رسانی در مدل داده‌ای احتیاج به تغییر در سیستم نرم‌افزاری دارد.
- تاکسونومی فقط یک الگو برای طبقه‌بندی مدل داده‌ای فراهم می‌کند، ولی هیچگونه ساختاری برای استدلال و تصمیم‌گیری سیستم نرم‌افزاری از روی نمونه‌های تاکسونومی فراهم نمی‌کند.

- اکثر زبان‌های حملات و امضای آنها فقط برای یک حوزه، محیط و یا یک سیستم بخصوص مناسب بوده و بین سیستم‌های ناهمگن قابل حمل نبوده و دارای معانی نامفهومی می‌باشند.

برای حل مشکلات فوق می‌توان به جای تاکسونومی از آنتولوژی استفاده کرد. تاکسونومی همواره جزئی از آنتولوژی می‌باشد، در واقع آنتولوژی علاوه بر مزایا و ویژگی‌های تاکسونومی دارای یکسری مزایای خاص خود نیز می‌باشد. استفاده از آنتولوژی برای حل مشکلات تشخیص نفوذ، روش بسیار مناسبی است. توانایی روش آنتولوژی زمانی نمایان می‌شود که بتوان رابطهٔ بین داده‌های جمع‌آوری شده را نشان داد و از این رابطه‌ها برای کاهش حجم نمایش داده‌ها و حمله‌های نوع خاص استفاده کرد. علاوه بر این با استفاده از یک روش نمایش مبتنی بر آنتولوژی برای تجزیه مدل داده‌ای، می‌توان با استفاده از منطق موجود در یک سیستم تشخیص نفوذ، یک نفوذ را شناسایی کرد. جداکردن مدل داده‌ای از منطق سیستم تشخیص نفوذ، سیستم‌های ناهمگون را قادر می‌سازد تا بدون هماهنگی و توافق قبلی داده‌های خود را به اشتراک بگذارند. در این روش‌ها یک نمونه از آنتولوژی در بین سیستم‌های تشخیص نفوذ در غالب یک جمله *DAML*، *RDF* و یا *OWL* به اشتراک گذاشته می‌شود.

برای اولین بار آقای *Victor Raskin* بحث استفاده از آنتولوژی در مباحث مربوط به امنیت اطلاعات را مطرح کرد[*Ras01*]. ایشان به این نکته اشاره داشتند که با استفاده از آنتولوژی می‌توان به یک ابزار طبقه‌بندی قوی برای مجموعهٔ نامحدودی از رویدادها دست یافت. با استفاده از مفهوم آنتولوژی می‌توان به سیستم کشف نفوذ این امکان را داد، که با استفاده از آنالیز و بررسی اطلاعات قبلی به اطلاعات و داده‌های جدید و مفیدی برسد. این روش به تشخیص رفتارهای مشکوک و شناسایی رفتارهایی که ممکن است، حمله و یا نفوذ باشند، کمک می‌کند.

با افزودن روش‌ها و مفاهیم وب معنایی، مانند روش‌های مبتنی بر آنتولوژی و توجه به محتوی و مضمون اطلاعات می‌توان عملکرد سیستم‌های تشخیص نفوذ را بهبود بخشید. از این روش‌ها می‌توان به طور گسترده در سیستم‌هایی که بر مبنای تشخیص رفتارهای مشکوک هستند، استفاده کرد. با توجه به اطلاعاتی که در رابطه با حملات و نفوذهای شناخته شده در سیستم وجود دارد و وقایع رخ داده در سیستم، رفتارهای مشکوکی که ممکن است نفوذ باشند، شناسایی می‌شوند. استفاده از آنتولوژی در سیستم‌های تشخیص نفوذ

کمک می‌کند تا عمل طبقه‌بندی از روی نمونه‌های دیتا مدل صورت بگیرد و همچنین به سیستم کمک می-کند تا از روی نمونه‌های طبقه‌بندی استدلال و تصمیم گیری کند.

۱– ۳– تحقیق انجام شده در این پایان نامه

در این پایان‌نامه هدف ارائه یک سیستم تشخیص نفوذ مبتنی بر آنتولوژی مستخرج از داده‌کاوی می‌باشد. از آنتولوژی "حملات کامپیوتری" برای انجام عمل تشخیص نفوذ استفاده می‌شود.

در اغلب کارهای انجام گرفته در زمینه سیستم‌های تشخیص نفوذ برای نمایش ویژگی‌های حوزه‌ی حملات و نفوذهای کامپیوتری از یک تاکسونومی استفاده می‌شود*[Und03]*. در این پروژه به جای تاکسونومی از آنتولوژی برای توصیف مفاهیم این حوزه و روابط بین آنها استفاده شده است. برای بدست آوردن این مفاهیم و ویژگی‌ها از روی مجموعه داده‌های آموزشی از تکنیک‌های مختلف داده‌کاوی مانند طبقه بندی، خوشه بندی وغیره استفاده شده است.

در مرحله‌ی بعدی تحقیق با بهره‌گیری از قوانین مستخرج از داده‌کاوی، یک آنتولوژی برای حملات مختلف، ویژگی‌های این حملات و روابط بین آنها ایجاد شده است. علاوه بر این اطلاعات، آنتولوژی بایستی شامل اطلاعات مربوط به وضعیت ارتباطات و اتصالات شبکه مورد نظر باشد تا با استفاده از آنها عمل جمع آوری داده‌های وضعیت کنونی شبکه و تصمیم گیری بر اساس آنها امکان پذیر باشد.

در آخرین فاز از این تحقیق نیز در یک سیستم همکارانه متشکل از عامل‌های مختلف (مبتنی بر میزبان، شبکه و...) از آنتولوژی بدست آمده برای تشخیص رفتارهای مشکوک و نفوذی در شبکه استفاده شده است. در این راستا علاوه بر بهره‌گیری از ویژگی‌های عامل‌های متحرک، استفاده از تکنیک‌های محاسبه، تشابه معنایی و تطابق آنتولوژی‌ها نیز کمک بسیار زیادی می‌نماید.

در واقع تحقیق انجام شده در این پایان‌نامه شامل دو بخش اصلی می‌باشد: بخش مربوط به طراحی آنتولوژی "حملات کامپیوتری" و بخش مربوط به طراحی و پیاده‌سازی سیستم تشخیص نفوذ پیشنهادی که در این پایان‌نامه با نام اختصاری[۱] *DIDMO* ذکر شده است. سیستم *DIDMO* وضعیت‌های رخ داده در شبکه را در دو

[۱] *Distributed Intrusion Detection Using Mined Ontology*

کلاس اصلی "حملات" و "نرمال" طبقه‌بندی می‌کند، هر کدام از این کلاس‌ها نیز به نوبه‌ی خود به زیر کلاس‌های دیگر تقسیم‌بندی می‌شوند. برای تست سیستم پیشنهادی و بررسی صحت عملکرد آنتولوژی طراحی شده نیز از مجموعه داده‌های آزمایشی موجود در پایگاه دادهٔ *NSL-KDD* استفاده شده است.

۱– ۴– نمای کلی پایان نامه

ساختار این پایان نامه به‌صورت زیر می‌باشد:

در فصل اول مقدمه‌ای بر تاریخچه‌ی تشخیص نفوذ در شبکه‌های کامپیوتری ارائه شده است. در این فصل برخی از مشکلات تاکسونومی در سیستم‌های تشخیص نفوذ بررسی، و راه‌کاری برای حل این مشکلات بیان می‌شود. یکی از راه‌کارهای حل این مشکلات بکارگیری آنتولوژی برای ارائه‌ی مدل داده‌ای سیستم تشخیص نفوذ می‌باشد که در این فصل به‌طور مختصر ذکر شده و در فصل چهارم به تفصیل توضیح داده می‌شود.

فصل دوم با تعریف مساله آغاز شده، تعاریف اولیه در تشخیص نفوذ، انواع سیستم‌های تشخیص نفوذ و نوع نگرش آنها به حملات و روشهای برخورد با حملات در شبکه کامپیوتری از جمله مباحثی است که در این فصل به آنها پرداخته شده است. در ادامه‌ی فصل دوم نیز به بررسی چند نمونه از معیارهای ارزیابی سیستم-های تشخیص نفوذ خواهیم پرداخت.

تکنیک‌های مختلف داده‌کاوی که تاکنون در راستای تشخیص نفوذ در شبکه‌های کامپیوتری ارائه شده است در فصل سوم مورد بررسی قرار گرفته‌اند. در این فصل روش‌های مختلف داده‌کاوی بیان شده و بیشتر بر روی روش‌های یادگیری ماشین و طبقه بندی تمرکز شده است.

فصل چهارم به دو بخش اصلی تقسیم می‌شود. بخش اول مروری بر وب معنایی و تعاریف و مفاهیم آن بوده و در ادامهٔ آن بحث وب معنایی در امنیت اطلاعات و تحقیقات انجام شده در زمینهٔ سیستم‌های تشخیص نفوذ مورد بررسی قرار گرفته است. در بخش دوم نحوهٔ طراحی آنتولوژی‌های مبتنی بر *OWL* و به طور کلی اجزا و ویژگی‌های این نوع آنتولوژی‌ها به تفصیل شرح داده شده است.

در فصل پنجم نیز نحوهٔ طراحی و استخراج آنتولوژی پیشنهادی در این پایان‌نامه "آنتولوژی حملات کامپیوتری" مورد بحث و بررسی قرار گرفته است، و نحوهٔ استخراج اطلاعات مورد نیاز برای طراحی آنتولوژی "حملات کامپیوتری" و فازهای مختلف از ایجاد آنتولوژی به تفصیل شرح داده شده است.

در فصل ششم ابتدا سیستم پیشنهادی معرفی شده، اجزای و نحوهٔ عملکرد آن شرح داده شده و در ادامهٔ فصل نحوهٔ استنتاج عامل مرکزی از روی آنتولوژی "حملات کامپیوتری" تشریح شده است. در انتهای فصل نیز آزمایشات و نتایج ارزیابی سیستم پیشنهادی مورد بحث و بررسی قرار گرفته و با چند روش دیگر مقایسه شده است. فصل آخر نیز به نتیجه‌گیری کلی و ارائه راه‌کارهایی برای فعالیت‌های آینده در این زمینه می‌پردازد.

فصل دوم

«پیش زمینه و بررسی کارهای مشابه»

۲–۱– مقدمه

این فصل با تعریف مساله و تعاریف اولیه در تشخیص نفوذ آغاز می‌شود، انواع سیستم‌های تشخیص نفوذ و نوع نگرش آن‌ها به حملات و روش‌های برخورد با حملات در شبکه کامپیوتری از جمله مباحثی است که در این بخش به آن پرداخته شده است. در ادامه‌ی این فصل به طبقه بندی سیستم‌های تشخیص نفوذ از دیدگاه‌های مختلف پرداخته، جزئیات هر نمونه و نمونه‌هایی از کارهای انجام گرفته در هر زمینه بیان شده است و درانتهای فصل نیز چند نمونه از معیارهای ارزیابی سیستم‌های تشخیص نفوذ بررسی خواهد شد.

۲–۲– تشخیص نفوذ

با فراگیر شدن کامپیوتر و گسترش شبکه‌های کامپیوتری در اجتماع امروز و به دلیل حملات و نفوذهایی که به شکل‌های مختلف به این شبکه‌ها صورت می‌گیرد، امنیت شبکه‌های کامپیوتری از حساسیت خاصی برخوردار گشته است. از آنجا که از نظر تکنیکی ایجاد سیستم‌های کامپیوتری (سخت افزار و نرم افزار) بدون نقاط ضعف و شکست امنیتی عملاً غیر ممکن است، کشف نفوذ در تحقیقات سیستم‌های کامپیوتری با اهمیت خاصی دنبال می‌شود.

نفوذ به مجموعه اقدامات غیر قانونی که صحت، محرمانه بودن و یا دسترسی به یک منبع را به خطر می‌اندازد اطلاق می‌گردد. نفوذها می‌توانند به دو دسته‌ی داخلی و خارجی تقسیم شوند. نفوذهای خارجی به آن دسته از نفوذها گفته می‌شود که توسط افراد مجاز و یا غیر مجاز از خارج شبکه به درون شبکه داخلی صورت می-گیرد و نفوذهای داخلی توسط افراد مجاز در سیستم و شبکه داخلی و از درون خود شبکه انجام می‌پذیرد.

نفوذگرها[۱] عموماً از عیوب نرم‌افزاری، شکستن کلمات رمز، استراق سمع ترافیک شبکه و نقاط ضعف طراحی در شبکه، سرویس‌ها و یا کامپیوترهای شبکه برای نفوذ به سیستم‌ها و شبکه‌های کامپیوتری بهره می‌برند. به عبارت دیگر یک نفوذگر کسی است که تلاش می‌کند به سیستم نفوذ کند و از آن سوء استفاده کند. البته کلمه سوءاستفاده پهنه وسیعی را در بر می‌گیرد و می‌تواند منعکس کننده عملی جدی مانند دزدیدن اطلاعات محرمانه و یا عمل کوچک و ناچیزی مثل دزدیدن آدرس *e-mail* جهت فرستادن *spam* باشد.

---

[۱] *Intruder*

به منظور مقابله با نفوذگران سیستم‌ها و شبکه‌های کامپیوتری، روش‌های متعددی تحت عنوان روش‌های تشخیص نفوذ ایجاد گردیده که عمل نظارت بر وقایع اتفاق افتاده در یک سیستم یا شبکه کامپیوتری را بر عهده دارند. هدف از تشخیص نفوذ کشف هرگونه استفاده غیر مجاز، سوءاستفاده و یا آسیب رساندن به سیستم‌ها و یا شبکه‌های کامپیوتری توسط هر دو دسته کاربران داخلی و خارجی است.

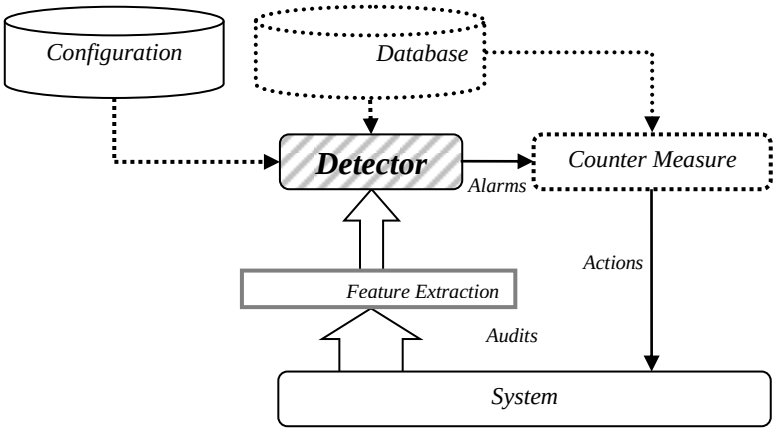
سیستم‌های کشف نفوذ به صورت سیستم‌های نرم‌افزاری و یا سخت‌افزاری ایجاد شده و هر کدام دارای مزایا و معایب خاص خود می‌باشند. سرعت و دقت از مزایای سیستم‌های سخت افزاری است و عدم شکست امنیت آن‌ها توسط نفوذگران قابلیت دیگر اینگونه سیستم‌هاست. اما استفاده آسان از نرم افزار و قابلیت انطباق پذیری در شرایط نرم‌افزاری و تفاوت سیستم عامل‌های مختلف، عمومیت بیشتری را در سیستم‌های نرم‌افزاری ارائه می‌کند و عموماً اینگونه سیستم‌ها انتخاب مناسب‌تری هستند.

به طور کلی یک سیستم تشخیص نفوذ فعالیت‌های محیطی را که در آن عمل می‌کند مانیتور کرده و سپس اطلاعات غیر ضروری را از اطلاعات بدست آمده حذف می‌کند، معمولاً یک سری خصیصه برای این اطلاعات جمع آوری شده استخراج می‌شود، آن‌گاه پس از ارزیابی فعالیت‌ها، احتمال وجود حمله بررسی می‌شود که این عمل توسط تشخیص دهنده انجام می‌گیرد. پس از تشخیص معمولاً سیستم واکنش مناسب را در قبال تهاجم تشخیص داده شده انجام می‌دهد. اغلب سیستم‌های تشخیص نفوذ همان‌طور که از اسم آن‌ها پیدا است، فقط حملات را تشخیص داده و هشدار می‌دهند و معمولاً هیچ عمل بازدارنده‌ای از آن‌ها صادر نمی‌شود، هر چند بعضی از انواع این سیستم‌ها علاوه بر تشخیص نفوذ واکنش‌های بازدارنده را نیز انجام می‌دهند. مهمترین بخش یک سیستم تشخیص نفوذ، تشخیص دهنده است که وظیفه اصلی وارسی اطلاعات جمع آوری شده را بر عهده دارد. شکل ۲- ۱ شمای کلی یک سیستم تشخیص نفوذ را بر اساس تعاریف ارائه شده نشان می‌دهد.

تاکنون روش‌های بسیار متنوعی در ساخت یک تشخیص دهنده به کار گرفته شده است، به عنوان مثال روش‌های داده‌کاوی *[lee95]*، نمودارهای گذرحالات *[Ilg95]*، روش‌های خوشه‌بندی *[Gua03]*و تکنیک‌های

[1] *Clustering*

طبقه‌بندی[Gom01] را می‌توان در این میان مشاهده کرد. نمونه‌های اشاره شده نمونه‌های اندکی از صدها روش به کار گرفته شده برای سیستم‌های تشخیص نفوذ است. در این مطالعه تلاش شده است تا از روش طبقه‌بندی الگو که از تکنیک‌های رایج داده‌کاوی است، برای ایجاد مدل داده‌ای یک سیستم تشخیص نفوذ استفاده شود.



شکل ۲- ۱: یک سیستم تشخیص نفوذ ساده

۲- ۳- طبقه بندی سیستم‌های تشخیص نفوذ

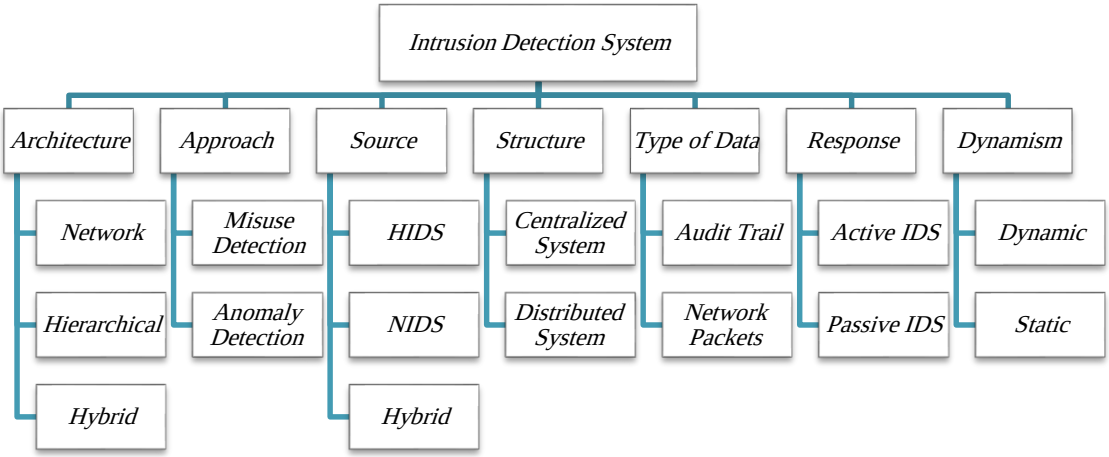
تاکنون تقسیم بندی‌های زیادی در زمینه‌ی انواع سیستم‌های تشخیص نفوذ صورت گرفته است با این وجود هنوز طبقه بندی‌ای[۱] که توسط تمام محققان پذیرفته شده باشد، وجود ندارد. بخشی از این موضوع به‌خاطر جوان بودن این شاخه نسبت به سایر شاخه‌های کامپیوتر بوده و بخش دیگر نیز به خاطر پیچیدگی‌های ذاتی این شاخه می‌باشد. در این بخش سعی شده سیستم‌های تشخیص نفوذ را بر اساس ویژگی‌های بارز آن‌ها طبقه‌بندی شود. شکل ۲- ۲ طبقه‌بندی ارائه شده‌ی ما را برای سیستم‌های تشخیص نفوذ نشان می‌دهد. در ادامه این طبقه بندی تشریح شده و ویژگی‌های گروه‌های مختلف طبقه‌بندی بررسی شده است.

[۱] *Classification*

[۲] *Taxonomy*

۲-۳-۱- معماری سیستم‌های تشخیص نفوذ

سیستم‌های تشخیص نفوذ به لحاظ نوع معماری به سه دسته‌ی سلسله مراتبی، شبکه‌ای و ترکیبی تقسیم می‌شوند. در یک معماری سلسله مراتبی[۱]گره‌های برگ نقش جمع آوری اطلاعات بدست آمده از شبکه و میزبان‌ها را به عهده دارند. این اطلاعات به گره‌ی پدر فرستاده می‌شود و گره‌ی پدر نیز نقش جمع آوری و به-هم پیوستن اطلاعات دریافتی از تمام فرزندان خود را به‌عهده دارد. جمع آوری، به‌هم پیوستن و حذف اطلاعات اضافی در تمام گره‌های داخلی انجام می‌شود و تمام این اطلاعات به گره‌ی ریشه می‌رسند. گره‌ی ریشه نقش کنترل و فرماندهی سیستم را به عهده داشته و پس از تشخیص نفوذ، رفتار مناسب با آن را انجام می‌دهد.



شکل ۲- ۲ : طبقه بندی سیستم‌های تشخیص نفوذ

در معماری شبکه‌ای[۲]اطلاعات می‌توانند از هر گره به تمام گره‌های دیگرمنتقل شده و عملیات کنترل و فرماندهی به عنوان مؤلفه تشخیص نفوذ در تمام گره‌های سیستم تحت حفاظت قرار می‌گیرد. معماری ترکیبی نیز در واقع شامل هر دو معماری سلسله مراتبی و شبکه‌ای می‌باشد.

[۱] *Hierarchical*

[۲] *Network*

۲- ۳- ۲- رویکرد تشخیص نفوذ

سیستم‌های تشخیص نفوذ به لحاظ رویکرد تشخیص نفوذ[۱]و تکنیک برخورد با نفوذها به سه دسته تقسیم می‌شوند *[Axe00]*.

۲- ۳- ۲- ۱- تشخیص سوء استفاده

در روش‌های تشخیص سوء استفاده[۲] الگوهای نفوذ از پیش ساخته شده به صورت قانون نگهداری می‌شوند، به طوری که هر الگو انواع متفاوتی از یک نفوذ خاص را در برگرفته و در صورت بروز چنین الگویی در سیستم نفوذ تشخیص داده می‌شود. به این روش‌ها، روش‌های مبتی بر امضاء[۳]نیز می‌گویند، زیرا معمولاً تشخیص دهنده دارای پایگاه داده‌ای از امضا یا الگوهای رخداد حمله است و سعی می‌کند با بررسی ترافیک شبکه الگو-های مشابه با آنچه را در پایگاه داده خود نگهداری می‌کند، بیابد.

سیستم‌های تشخیص نفوذ مبتنی بر امضاء ویا تشخیص سوءاستفاده یکی از روش‌هایی است که بیشتر استفاده می‌شود و دارای دقت تشخیص بالایی است. هنگامی‌که یک حمله جدید رخ می‌دهد دقیقاً مورد تجزیه و تحلیل قرار می‌گیرد و امضای آن مشخص می‌شود. با مشخص کردن الگوی حمله می‌توان در مقابل آن حمله دفاع نمود. بر مبنای این روش، پایگاه داده‌ی الگوهای حمله با شناختن الگوهای جدید بایستی بروز شود. این روش مفیدی برای حمله‌های شناخته شده می‌باشد و دارای نرخ خطای مثبت[۴]کمتری است. یکی از معایب این روش این است که قادر نیست حمله‌های جدید را بشناسد. علاوه بر این روش، تکنیک‌های مختلف دیگری برای پیاده‌سازی سیستم تشخیص نفوذ بکار گرفته شده است که قادر به شاختن حمله‌های جدید نیزمی-باشند.

در *[And95]* و *[Mou95]* دو روش پیاده‌سازی سیستم سوء استفاده بکار برده شده است که از قانون‌ها[۵]برای پیاده‌سازی استفاده کرده‌اند. در *[Lin99]* یک مدل از انتقال حالت[۱]برای پیاده‌سازی سیستم تشخیص نفوذ

---

[۱] *Intrusion Detection Approach*

[۲] *Misuse Detection*

[۳] *Signature Based*

[۴] *False Positive*

[۵] *Rules*

سوء استفاده، استفاده شده است. در *[Ilg95]* و *[Kum95]* آتوماتای رنگی پتری[۲]برای پیاده‌سازی سیستم تشخیص نفوذ سوء استفاده، بکار گرفته شده است.

۲- ۳- ۲- ۲- تشخیص رفتار غیر عادی

در روش‌های تشخیص رفتار غیرعادی[۳]که بر اساس رفتار نرمال سیستم بنا شده است، در یک بازهٔ زمانی خاص و مطمئن از این جهت که هیچگونه نفوذی در این بازه صورت نگرفته، اقدام به ایجاد نماهائی رفتار عادی کاربران کرده و در صورتی که رفتار سیستم در زمان آزمایش از این الگوها تبعیت نکند به صورت یک نفوذ احتمالی در نظر گرفته می‌شود. از مزایای این روش‌ها این است که قادرند حملات و فعالیت‌های نفوذی جدید که هیچ آموزشی برای آنها ندیده‌اند را شناسایی کنند، زیرا همانطور که گفته شد این روش‌ها به دنبال الگوهایی می‌گردند که به اندازه کافی متفاوت از رفتارهای نرمال شبکه بوده و بدین ترتیب قادر به تشخیص حملات جدید می‌باشند. با این وجود ایجاد یک سیستم تشخیص نفوذ بر اساس این روش تشخیص آنامولی یا رفتارهای غیرعادی همیشه کار آسانی نیست و این روش‌ها از دقت روش‌های تشخیص سوء استفاده برخوردار نیستند. در این روش‌ها بصورت دقیق نمی‌توان مشخص نمود که آیا الگوی تشخیص داده شده واقعا نرمال است یا غیر نرمال، بنابراین نرخ خطای مثبت برای سیستم تشخیص رفتار غیرعادی بالا می‌باشد.

روش‌های تشخیص نفوذ گوناگونی که بر این اساس کار می‌کنند تاکنون ارائه شده است که از معروف‌ترین آن-ها می‌توان به روش ارائه شده توسط *Denning* اشاره نمود *[Den87]*، که بر اساس معیارهای آماری داده‌ها کار می‌کند.

۲- ۳- ۲- ۳- تشخیص ترکیبی

سیستم‌های تشخیص نفوذ سوءاستفاده یکسری سناریوی نفوذ شناخته شده را مدل کرده و در سیستم مانیتور می‌کند، این سیستم‌ها قابلیت شناخت نفوذهای جدید را ندارند. سیستم‌های تشخیص نفوذ رفتار غیر عادی

---

[۱] *State Transition*

[۲] *Colored Petri Automata*

[۳] *Anomaly Detection*

[۴] *Profile*

قادر به تشخیص نفوذهای جدید می‌باشند. با توجه به گسترش بیش از حد حمله‌های کامپیوتری سیستم‌های تشخیص نفوذ سوء استفاده که دارای سرعت بالایی در پیاده‌سازی می‌باشند کارائی پایینی در برابر نفوذهای شناخته نشده دارند. بنابراین برای اینکه بتوان حمله‌های جدید را تشخیص داد و دقت سیستم تشخیص نفوذ را بالا برد نیاز به استفاده از سیستم‌های ترکیبی[۱]رفتار غیرعادی به همراه سیستم تشخیص نفوذ می‌باشد تا بتواند نفوذهای شناخته شده و جدید را بشناسد.

به عبارت دیگر تشخیص در این روش هم با استفاده از فعالیت‌های نرمال سیستم و فعالیت‌های نفوذی نفوذگر انجام می‌گیرد. از معروف‌ترین این روش‌ها می‌توان به روش ارائه شده در *[Lee99]* اشاره نمود که از قوانین *RIPPER* برای تشخیص نفوذ استفاده می‌کند. در *[Bur04]* از ترکیبی از مدل مبتنی بر حالت خاص (برای استخراج ترافیک نرمال شبکه) و نیز استفاده از الگوریتم کلاسترینگ افزایشی[۲](برای تشخیص نفوذهای جدید) برای پیاده سازی سیستم تشخیص نفوذ استفاده شده است.

۲– ۳– ۳– منبع تشخیص نفوذ

علاوه بر تقسیم بندی‌های فوق به طور کلی سیستم‌های تشخیص نفوذ از نقطه نظر منبعی که تشخیص نفوذ روی آن صورت می‌گیرد نیز به دو دسته تقسیم می‌شوند *[Der99]*.

۲– ۳– ۳– ۱– تشخیص نفوذ بر اساس مدل میزبان

در روش مبتنی بر میزبان، تشخیص نفوذ در یک سیستم منفرد مد نظر بوده و معمولا این روش‌ها بر اساس فعالیت‌های کاربر سیستم شامل فراخوانی‌های سیستمی و غیره می‌باشد و می‌تواند به هر دو صورت تشخیص نفوذ مبتنی بر تشخیص سوءاستفاده و یا تشخیص رفتار غیرعادی انجام شود.

---

[۱] *Compound Detection*

[۲] *Incremental Clustering*

۲- ۳- ۳- ۲- تشخیص نفوذ بر اساس ترافیک شبکه

در تشخیص نفوذ مبتنی بر ترافیک شبکه حمله به کل ساختار شبکه و یا هر یک از میزبان‌های شبکه، می-تواند سیستم تشخیص نفوذ را برای اعلام نفوذ فعال کند. این روش نیز می‌تواند به صورت تشخیص سوء-استفاده و یا تشخیص رفتار غیرعادی اعمال شود.

تفاوت این دو دسته در منبع داده‌ای است که سیستم تشخیص نفوذ برای جمع‌آوری اطلاعات از آن بهره می-برد. در تشخیص نفوذ مبتنی بر مدل میزبان، منبع داده از اطلاعات یک کامپیوتر استفاده می‌کند، به این صورت که یک عامل[۱]هوشمند بر روی میزبان نظارت شده نصب می‌گردد. و جنبه‌های متفاوتی از امنیت میزبان از قبیل فایل‌های رخدادهای سیستم عامل، برنامه‌های کاربردی و غیره را در نظر می‌گیرد. در این صورت منبع داده‌ای این سیستم‌ها، رکوردهای ردیابی سیستم عامل، رخدادهای سیستمی و اطلاعات مربوط به برنامه‌های کاربردی و غیره را در نظر می‌گیرد. از نمونه سیستم‌هایی که بر این اساس پیاده‌سازی شده‌اند می‌توان به *[Dow90]* و *[And95]* اشاره کرد.

در تشخیص نفوذ مبتنی بر شبکه از ترافیک شبکه به عنوان منبع اصلی اطلاعات استفاده می‌شود در این سیستم‌ها معمولاً کارت واسطه شبکه در وضعیت *Promiscuous* قرار داده می‌شود به این ترتیب سیستم تشخیص نفوذ قادر است تمامی ترافیک شبکه که از سگمنت شبکه‌اش عبور می‌کند را ملاحظه کند.

در *[Muk02]* نویسنده معتقد است که سیستم‌های تشخیص نفوذ به دو دسته تقسیم می‌شوند. مبتنی بر میزبان و مبتنی بر شبکه. این نوع سیستم، تشخیص نفوذ را به‌این صورت تعریف می‌کند: "سیستم تشخیص نفوذ مبتنی بر میزبان تمام وقایع موجود بر روی یک میزبان را مانیتور می‌کند و مطمئن است که هیچ گونه سیاست امنیتی نقض نمی‌شود. سیستم تشخیص نفوذ مبتنی بر شبکه فعالیت کل شبکه را مانیتور می‌کند و ترافیک شبکه را مورد تجزیه و تحلیل قرار می‌دهد".

---

[۱] *Agent*

۲– ۳– ۴– ساختار سیستم تشخیص نفوذ

از زمانی که مفهوم سیستم‌های تشخیص نفوذ برای اولین بار *(Anderson, 1980)* مطرح شد تاکنون تعداد زیادی سیستم تشخیص نفوذ برای سیستم‌های متمرکز، طراحی و پیاده‌سازی شده است. در یک سیستم تشخیص نفوذ متمرکز عمل آنالیز داده‌ها در محل‌های مشخص و ثابتی انجام می‌پذیرد که تعداد این محل‌ها ثابت بوده و به تعداد میزبان‌های موجود در سیستم تحت نظارت وابسته نمی‌باشد.

در یک سیستم توزیع شده، سیستم تشخیص نفوذ بدون افزودن بار قابل ملاحظه بر روی سیستم‌های تحت نظارت و شبکه، بایستی حجم زیادی از داده‌ها را آنالیز نماید. این داده‌ها نیز از سراسر سیستم محاسباتی جمع آوری می‌شوند. برای این هدف از تکنولوژی عامل‌ها استفاده می‌شود و عامل‌های سیار[۱]در سراسر شبکه توزیع شده و عمل جمع آوری داده‌ها را به صورت خود مختار و کاملا هوشمندانه انجام می‌دهند. سیستم‌های تشخیص نفوذ مبتنی بر عامل‌های سیار، نقایص و کاستی‌های سیستم‌های متمرکز را با بهره‌گیری از الگوی عامل‌های سیار برای ایجاد همبستگی[۲]توزیع شده از بین می‌برد.

۲– ۳– ۴– ۱– عامل‌ها و سیستم‌های تشخیص نفوذ توزیع شده

سیستم تشخیص نفوذ توزیع شده مسئول محافظت محیط کل شبکه در برابر نفوذ می‌باشد. این امر نیاز به اطلاعات کامل وضعیت سیستم و مانیتور کردن قسمت‌های مختلف و ارتباط بین آنها دارد. تکنولوژی عامل نقش حیاتی در این زمینه ایفا می‌کند. شبکه یک ساختار برای سیستم‌های توزیع شده بوده و بنابراین عامل یک انتخاب طبیعی برای این دیدگاه می‌باشد. بدست آوردن اطلاعات از قسمت‌های مختلف شبکه، پردازش آن‌ها و پاسخ به درخواست آن‌ها، بصورت مستقل می‌تواند از طریق عامل‌ها بدست آید. سیستم‌های تشخیص نفوذ توزیع شده دارای توانایی دستیابی به روترهای شبکه و پیکربندی آن‌ها بمنظور انجام عملیات مورد نظر می‌باشند. مثلا می‌توانند از روتر درخواست کنند که پایانه و یا شبکه‌ای را که حاوی نفوذ می‌باشد قطع نماید.

در این زمینه کارهای زیادی انجام شده است مثلاً در *[Sim04]* یک سیستم تشخیص نفوذ مبتنی بر عامل‌ها ارائه شده که در آن سعی بر بهبود سرعت توسعه و تغییر سیستم تشخیص نفوذ شده است. در *[Zha01]* نیز

---

[۱] *Mobile Agent*

[۲] *Correlation*

یک مدل مبتنی بر عامل برای سیستم تشخیص نفوذ اعلام شده است. در *[Luo03]* از تکنولوژی عامل‌های متحرک برای پیاده سازی سیستم تشخیص نفوذ استفاده شده است. در *[Ram04]* نیز یک مدل مبتنی بر عامل‌ها که از مفهوم همسایگی استفاده نموده ارائه شده است.

۲- ۳- ۵- نوع داده

تمام پروژه‌هایی که بر روی توسعه یا آزمایش یک سیستم تشخیص نفوذ تمرکز می‌کنند از داده‌هایی برای این کار استفاده می‌کنند. اکثر تحقیقات بر روی جمع آوری داده‌ها تمرکز نمی‌کنند ولی وابستگی شدیدی با داده-های مصرفی‌شان، دارند*[Sod04]*. انواع داده‌های جمع آوری شده از منابع مختلف شامل اثر رسیدگی[۱] و بسته-های شبکه[۲] می‌باشند.

- اثر رسیدگی در واقع همان وقایع ثبت[۳]شده‌ی رخدادها در محیط شبکه می‌باشند.
- بسته‌های شبکه نیز واحدهایی از اطلاعات بوده که در محیط شبکه از یک کامپیوتر به کامپیوتر دیگر منتقل می‌شوند.

۲- ۳- ۶- مکانیزم پاسخ

بر اساس مکانیزم پاسخ نیز سیستم‌های تشخیص نفوذ به دو نوع فعال[۴] و غیر فعال[۵] تقسیم‌بندی می-شوند*[Bac02]*. در سیستم‌های فعال پس از تشخیص نفوذ، عمل خاصی برای جلوگیری از آن انجام می‌پذیرد. مثلا ترافیک شبکه را با استفاده از دیواره آتش بلاک می‌کند. ولی در سیستم‌های غیر فعال وجود یک بخش دیگر برای انجام واکنش ضروری می‌باشد. مثلا این سیستم‌ها مدیر یا مسئول امنیتی شبکه را از وقوع یک نفوذ مطلع می‌کنند و هیچ وظیفه‌ای در قبال انجام واکنش مستقیم برای جلوگیری از نفوذ به عهده ندارند.

---

[۱] *Audit Trail*
[۲] *Network Packet*
[۳] *Log*
[۴] *Active*
[۵] *Passive*

۲– ۳– ۷– پویایی سیستم

از دیدگاه پویایی نیز سیستم‌های تشخیص نفوذ را می‌توان به دو نوع ایستا[۱]و پویا[۲]تقسیم کرد. که در حالت ایستا سیستم در ابتدا مدل می‌شود سپس بصورت ثابت برای شناخت نفوذ استفاده می‌شود. در حالت پویا سیستم علاوه بر اطلاعات قبلی که یاد گرفته است، از اطلاعات جدید نیز برای شناخت نفوذ استفاده می‌کند و خود را بصورت پویا با اطلاعات جدید به‌روز می‌کند.

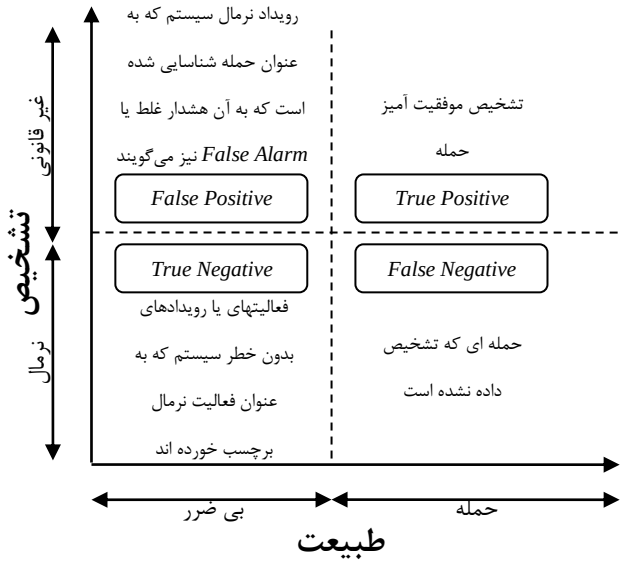
۲– ۴– معیارهای ارزیابی سیستم تشخیص نفوذ

از ویژگی‌های بسیار مهم یک سیستم تشخیص نفوذ دقت در تشخیص و شناخت نفوذهای اتفاق افتاده و عدم تشخیص غلط رفتارهای نرمال به عنوان یک رفتار غیرعادی می‌باشد. ضعف یک سیستم کشف نفوذ در هر یک از این دو ویژگی باعث می‌شود تأثیر منفی در کارآیی سیستم اصلی داشته باشیم تا جایی که عدم حضور سیستم تشخیص نفوذ در چنین حالتی منطقاً بر حضور آن ترجیح داده می‌شود. چرا که هشدارهای غلط مسئول امنیتی سیستم را از حضور سیستم کشف نفوذ خسته کرده وهمچنین هشدارهایی که به دلیل ضعف سیستم اعلام نمی‌شوند، امنیت سیستم کشف نفوذ را زیر سئوال خواهند برد. به طور کلی واکنش‌های یک سیستم تشخیص نفوذ در قبال رویدادهای مختلف سیستم را می‌توان به چهار دسته ارائه شده در شکل ۲- ۳ تقسیم نمود.

---

[۱] *Static*

[۲] *Dynamic*



شکل ۲- ۳: وضعیت‌های ممکن در قبال واکنشهای یک سیستم تشخیص نفوذ

True Positive (TP): تعداد رکوردهای نفوذ که بصورت نفوذ کلاس بندی شده اند.

True Negative (TN): تعداد رکوردهای نرمال که بصورت نرمال کلاس بندی شده اند.

False positive (FP): تعداد رکوردهای نرمال که بصورت نفوذ کلاس بندی شده اند.

False Negative (FN): تعداد رکوردهای نفوذ که بصورت نرمال کلاس بندی شده اند.

معیارهای استاندارد برای ارزیابی سیستم تشخیص نفوذ شامل نرخ تشخیص[۱] نرخ هشدار غلط[۲] مقایسه بین نرخ تشخیص و نرخ هشدار غلط، کارائی (سرعت پردازش + انتشار + عکس العمل) و تحمل خطا[۳] در مقابل حمله، ترمیم و انهدام[۴] می‌باشد.

نرخ تشخیص برحسب تعداد حمله‌هایی که درست شناخته می‌شوند و تعداد کل حمله‌ها محاسبه می‌شود در حالی‌که نرخ هشدار اشتباه مثبت بر حسب تعداد کل رکوردهای نرمال که اشتباهاً بعنوان حمله شناخته شده-اند محاسبه می‌شود. این دو پارامتر خوبی برای سنجش کارائی بوده زیرا درصدی از حمله‌ها را که سیستم

---

[۱] *Detection Rate*

[۲] *False Positive Rate*

[۳] *Fault Tolerance*

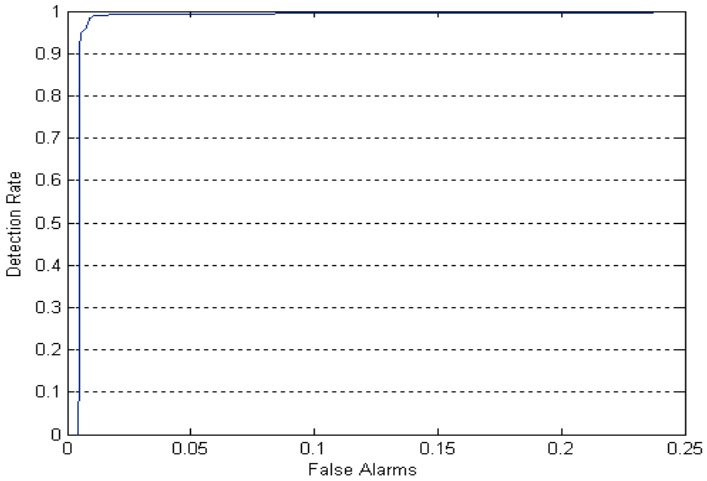
[۴] *Subversion*

می‌تواند تشخیص دهد و نیز تعداد رکوردهای نرمالی که به‌عنوان حمله شناخته شده‌اند را مشخص می‌کنند. بنابراین می‌توان نوشت :

Detection Rate (DTR) = (TP)/ (TP+FN)
False Positive Rate (FPR) = (FP)/ (TN+FP)
Overall Accuracy (OA) = (TP+TN)/ (TP+TN+FP+FN)

یکی دیگر از ابزارهایی که برای ارزیابی سیستم تشخیص نفوذ بکار گرفته می‌شود نمودار *ROC*[۱] بوده که بصورت نرخ تشخیص در برابر نرخ خطای غلط تعریف می‌شود که محور افقی آن نرخ خطای غلط و محور عمودی آن نرخ تشخیص می‌باشد. هر چقدر انحنای این منحنی نزدیکتر به گوشه بالای سمت چپ باشد کارائی سیستم بهتر است.

به عبارت دیگر منحنی *ROC* نشان می‌دهد که تغییرات پارامترهای سیستم چگونه بر معیارهای ارزیابی سیستم تاثیر می‌گذارد. منحنی *ROC* می‌تواند مشخص کند که طبقه‌بند چه موقع کارایی خوبی دارد و برای هر طبقه‌بند بالاترین نقطه سمت چپ نمودار، تشخیص بهینه را بر اساس نرخ تشخیص و نرخ هشدار غلط نشان می‌دهد. اگر منحنی *ROC* برای طبقه‌بند *A* در تمامی قسمت‌های نمودار بالای منحنی طبقه‌بند *B* قرار گیرد آنگاه می‌توان ادعا کرد که طبقه‌بند *A* از طبقه‌بند *B* بهتر است و هر چه منحنی *ROC* برای یک سیستم طبقه‌بندی به گوشه سمت چپ و بالای نمودار نزدیک‌تر باشد طبقه‌بند مناسب‌تری است.



شکل ۲- ۴: منحنی *ROC*

[۱] *Receiver Operating Characteristic*

معیار دیگری که برای ارزیابی الگوریتم‌های طبقه‌بندی در نظر گرفته شده است، معیار هزینه برای نمونه[۱] است. در این پایان‌نامه به این معیار با نام اختصاری *CPE* اشاره می‌شود. *CPE* به صورت زیر محاسبه می‌شود:

$$CPE = \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{m} CM(i,j) * C(i,j)$$

که در آن *CM* و *C* به ترتیب ماتریس‌های برهم‌ریختگی[۲] و هزینه[۳]، *N* تعداد کل نمونه‌های آزمایش و *m* تعداد کلاس‌های موجود در عملیات طبقه‌بندی است. ماتریس برهم ریختگی یک ماتریس مربعی است که هر ستون آن به یک کلاس پیش‌بینی شده[۴] و هر سطر آن به یک کلاس واقعی[۵] اختصاص دارد. هر درایه در سطر *i* و ستون *j*، *CM(i,j)*، تعداد نمونه‌هایی که بدرستی طبقه‌بندی نشده‌اند را نشان می‌دهد، که در اصل به کلاس *i* متعلق بوده است و در کلاس *j* طبقه‌بندی شده است. درایه‌های روی قطر اصلی ماتریس تعداد نمونه‌هایی که بدرستی طبقه‌بندی شده‌اند، را مشخص می‌کند.

ماتریس هزینه از نظر ساختار شبیه به ماتریس برهم‌ریختگی است، با این تفاوت که درایه *C(i,j)* در این ماتریس هزینه جریمه برای طبقه‌بندی نادرست یک الگو از کلاس *i* در کلاس *j* را مشخص می‌کند. بنابراین درایه‌های قطر اصلی ماتریس *C* همواره دارای مقدار صفر هستند، زیرا قطر اصلی بیانگر طبقه‌بندی درست نمونه‌ها می‌باشد. هر چه مقدار *CPE* برای یک طبقه‌بندی کننده پایین‌تر باشد، بیان‌گر این مطلب است، که آن طبقه‌بندی کننده بهتر عمل می‌کند. در جدول ۲- ۱ ماتریس هزینه‌ای که در مسابقه یادگیری طبقه‌بندی کننده‌ها *KDD'99* به کار گرفته شده است *[Sab03]*، نشان داده شده است.

جدول ۲- ۱: ماتریس هزینه‌ی استفاده‌شده در مسابقه یادگیری طبقه‌بندی کننده‌های *KDD'99*

		Predicted				
		Normal	*PROBE*	*DoS*	*U2R*	*R2L*
Actual	*Normal*	*0*	*1*	*2*	*2*	*2*
	PROBE	*1*	*0*	*2*	*2*	*2*
	DoS	*2*	*1*	*0*	*2*	*2*
	U2R	*3*	*2*	*2*	*0*	*2*

[۱] *Cost Per Example*
[۲] *Confusion Matrix*
[۳] *Cost Matrix*
[۴] *Predicted Class*
[۵] *Actual Class*

	R2L	*4*	*2*	*2*	*2*	*0*

۲–۵– خلاصه

این فصل مروری بود بر برخی سیستم‌های کشف نفودی که تاکنون مطرح وپیاده‌سازی شده است ودر آن انواع قالب‌های کاری و رویکردهای مختلف آنها بررسی، و مزایا ومعایب هر یک به اختصار مطرح شد. این فصل با تعریف مساله آغاز شد و در ادامه به تعاریف اولیه در تشخیص نفوذ، انواع سیستم‌های تشخیص نفوذ و نوع نگرش آنها به حملات، روشهای برخورد با حملات در شبکه کامپیوتری پرداخته شد. به طبقه بندی سیستم-های تشخیص نفوذ از دیدگاه‌های مختلف پرداخته، ویژگی‌های آنها و برخی از کارهای انجام گرفته در هر زمینه را بیان شد. در نهایت نیز به بررسی معیارهای ارزیابی سیستم‌های تشخیص نفوذ پرداخته شد.

بررسی متون و مطالعه روش‌های ارائه شده در زمینه تشخیص نفوذ حاکی از اهمیت روش‌های مختلف داده-کاوی در این زمینه تحقیقاتی است. تاکنون روش‌های داده‌کاوی گوناگونی برای تشخیص نفوذ ارائه شده است که سیستم‌های یادگیری ماشین در این میان نقش بسزایی را بر عهده داشته و بررسی‌ها از تمایل هر چه بیشتر برای استفاده از این روش‌ها در این زمینه خبر می‌دهد. در فصل بعدی روش‌های مختلف داده‌کاوی برای حل مساله تشخیص نفوذ معرفی شده و در ادامه نیز روش *RIPPER* به عنوان بخشی از چهارچوب ارائه شده در این پایان‌نامه به‌طور کامل بررسی خواهد شد.

فصل سوم

«تکنیک‌های داده‌کاوی در تشخیص نفوذ»

۳-۱- مقدمه

همانطور که پیشتر نیز گفته شد نفوذ به مجموعه اقدامات غیر قانونی که صحت و محرمانه بودن و یا دسترسی به یک منبع را به خطر می‌اندازد، اطلاق می‌گردد. به منظور مقابله با این اعمال روش‌های متعددی تحت عنوان روش‌های تشخیص نفوذ ایجاد گردیده که عمل نظارت بر وقایع رخ داده در یک شبکه کامپیوتری را بر عهده دارند. روش عمده برای سیستم‌های تشخیص نفوذ این است که زمانی که مکانیزم رسیدگی[۱] برای قسمتی از سیستم فعال شد، شواهد مختلفی برای توصیف رفتار نرمال و غیرنرمال در فایل رسیدگی یافت شود. به خاطر حجم اطلاعات جمع آوری شده در فایل رسیدگی و همچنین تعداد صفات فراوان (صفات فایل رسیدگی) روش‌های موثر و هوشمند برای تجزیه و تحلیل این داده‌ها و استخراج رفتار نرمال و غیرنرمال سیستم لازم است. چندین نوع الگوریتم داده‌کاوی در بانک‌های اطلاعات وجود دارد که می‌توان آنها را برای کاوش در فایل رسیدگی و یا فایل ثبت ترافیک شبکه بکار گرفت. امروزه کاربرد و توسعه تکنیک‌های داده-کاوی (بینایی ماشین) در سیستم‌های تشخیص نفوذ اهمیت زیادی پیدا کرده است. ما در این بخش از نوشتار تمرکز بر روی تکنیک‌های داده‌کاوی بکار گرفته شده در سیستمهای تشخیص نفوذ قرار گرفته و در مورد نقاط ضعف و قوت این تکنیک‌ها بحث می‌شود. یکی از اصلی‌ترین اعمال مورد توجه برای پیاده‌سازی در این تحقیق طبقه‌بندی رفتارهای موجود در یک شبکه به رفتارهای نرمال یا رفتارهای نفوذی و یا حتی انواع مختلف رفتارهای نفوذی با استفاده از الگوریتم *RIPPER* بوده که در ادامه‌ی این تحقیق به طور کامل تشریح می‌شود.

۳-۲- داده‌کاوی

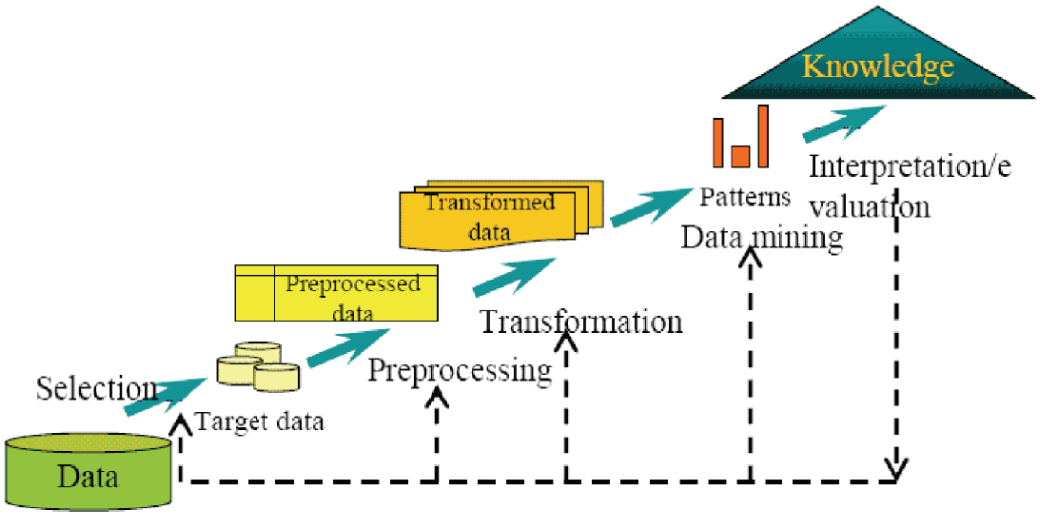
عبارت داده‌کاوی مترادف با یکی از عبارت‌های استخراج دانش، برداشت اطلاعات، وارسی داده‌ها و حتی لایروبی کردن داده‌هاست که در حقیقت کشف دانش[۲](*KD*) را توصیف می‌کند. بنابراین ایده‌ای که مبنای داده‌کاوی است یک فرآیند با اهمیت از شناخت الگوهای بالقوه مفید، تازه و درنهایت قابل درک در داده‌هاست. داده‌کاوی کاربرد سطح بالای تکنیک‌ها و ابزارها برای معرفی و تحلیل داده‌های تصمیم گیرنده‌گان است. داده-

---

[۱] *Audit*

[۲] *Knowledge Discovery*

کاوی در واقع فرایند جستجوی اتوماتیک در بین حجم زیادی از داده‌ها برای ایجاد الگوها با استفاده از قوانین همبستگی[۱] و کشف دانش می‌باشد. در شکل ۳- ۱ فرایند کشف دانش نشان داده شده است.



شکل ۳- ۱ : فرایند تبدیل داده‌های خام به دانش مفید

داده‌کاوی یکی از جدیدترین تکنیک‌ها در علوم کامپیوتر بوده و از تکنیک‌های قدیمی‌تری برای آمارگیری، بازیابی اطلاعات، یادگیری ماشین[۲] و تشخیص الگو استفاده می‌کند. در اینجا چند نمونه از کاربردهای داده-کاوی در پروژه‌های مربوط به تشخیص نفوذ ذکر می‌شود:

- حذف فعالیت‌های نرمال از داده‌های هشدار که باعث می‌شود تحلیل‌گر بر روی داده‌های هشدار تمرکز بیشتری داشته باشد.
- تشخیص تولید کننده‌های هشدار غلط و امضای غلط سنسورهای سیستم
- پیدا کردن رفتارهای غیرعادی که در واقع مربوط به یک حمله واقعی نمی‌باشند.

برای انجام این اعمال داده‌کاو می‌تواند از تکنیک‌های زیر استفاده کند:

- خلاصه سازی داده‌ها به وسیله تکنیک‌های آماری
- تجسم[۳]: ارائه یک خلاصه گرافیکی و قابل مشاهده از داده

[۱] *Association Rules*
[۲] *Machine Learning*
[۳] *Visualization*

- خوشه‌بندی[۱] داده‌ها به گروه‌های طبیعی
- کشف قوانین همبستگی: تعریف فعالیت‌های نرمال و ایجاد امکان کشف رفتارهای غیرنرمال
- طبقه بندی:[۲] پیش بینی تعلق یک نمونه‌ی خاص به گروه‌های از پیش تعریف شده

۳– ۳– داده‌کاوی در سیستم‌های تشخیص نفوذ

تکنیک‌های مختلف داده‌کاوی بر اساس معیارهایی مانند توابع گوناگون مدل‌سازی، میزان کارایی و الگوریتم-های گوناگون از هم متمایز می‌شوند. مدل‌های مستخرج از داده‌کاوی بایستی در قالب‌های گوناگون نمایش داده شوند. از رایج‌ترین قالب‌های نمایش مدل در داده‌کاوی می‌توان قوانین، درخت‌های تصمیم گیری، توابع خطی و غیرخطی (مانند شبکه‌های عصبی)، روش‌های مبتنی بر نمونه[۳] و مدل‌های احتمال را نام برد*[Fay96]*. در ادامه برخی از تکنیک‌های داده‌کاوی که تاکنون در سیستم‌های تشخیص نفوذ به‌کار گرفته شده‌اند مورد بررسی قرار گرفته است.

۳– ۳– ۱– انتخاب خصیصه‌ها

تکنیک انتخاب خصیصه‌ها[۴] که به انتخاب زیر مجموعه‌ها یا انتخاب متغیر معروف است فرایندی است که غالباً در روش‌های یادگیری ماشین استفاده می‌شود و کاربرد آن در مواقعی است که می‌خواهیم از بین مجموعه‌ای از خصیصه‌ها، زیر مجموعه‌ای از آنها را برای الگوریتم یادگیری انتخاب نماییم. استفاده از این تکنیک هنگامی که استفاده از تمام خصیصه‌های ممکن میسر نباشد و یا در مسائل مربوط به ارزیابی نمونه‌های داده‌ای محدود شده امری ضروری می‌باشد.

انتخاب خصیصه‌های داده‌ای برای افزایش کارایی روش استفاده شده امری بسیار حیاتی می‌باشد. محققان از روال‌های تحلیلی مختلفی برای انتخاب خصایص از داده‌های متراکم استفاده می‌کنند و خصایصی را انتخاب

---

[۱] *Clustering*
[۲] *Clasification*
[۳] *Instance Based*
[۴] *Feature Selection*

می‌کنند که باعث افزایش کارایی تکنیک داده‌کاوی می‌شوند. سه روش کلی برای انتخاب خصایص مرتبط با داده‌کاوی وجود دارد:

- انتخاب پیش رونده: در هر مرحله خصیصه‌ای که بیشترین ارتباط را دارد، انتخاب می‌شود.
- انتخاب پس رونده: در هر مرحله خصیصه‌ای که کمترین ارتباط را دارد، انتخاب و حذف می‌شود.
- روش ترکیبی: ترکیب هر دو روش پیش‌رونده و پس‌رونده.

۳- ۳- ۲- یادگیری ماشین

به عنوان یکی از شاخه‌های وسیع و پرکاربرد هوش مصنوعی، یادگیری ماشین[۱]به تنظیم و اکتشاف شیوه‌ها و الگوریتم‌هایی می‌پردازد که بر اساس آن رایانه‌ها و سامانه‌ها توانایی تعلّم و یادگیری می‌یابند. یادگیری ماشین در واقع به بررسی الگوریتم‌های کامپیوتری مختلفی اختصاص دارد که به طور اتوماتیک در طول تمرین و آزمایش بهبود یافته و آموزش می‌بینند.

در این زمینه می‌توان کاربردهایی مانند برنامه‌های داده‌کاوی‌ای که قوانین کلی[۲]را از بین مجموعه عظیمی از داده‌ها استخراج می‌کنند و یا سیستم‌های تصفیه[۳]اطلاعات که بطور خودکار علایق کاربران را یاد می‌گیرند، نام برد. برای یادگیری الگوی داده‌هایی که راجع به آنها دانشی در دسترس نیست، تکنیک‌های یادگیری ماشین بسیار بهتر از تکنیک‌های آماری[۴]عمل می‌نمایند. طبقه‌بندی و خوشه‌بندی دو مورد از پرکاربردترین تکنیک‌های یادگیری ماشین بوده که در ادامه این تکنیک‌ها بررسی شده است.

۳- ۳- ۲- ۱- تکنیک‌های طبقه بندی

در یک تکنیک طبقه‌بندی روش کلی کار بدین صورت است که یک نمونه از مجموعه داده‌ای گرفته و به یکی از کلاس‌های از پیش تعیین شده نسبت داده می‌شود. در این حالت کلاس‌های از پیش تعیین شده تعیین می‌کنند که چه چیزی را باید یاد بگیریم. در واقع در این حالت عمل یادگیری از طریق داده‌های آزمایشی که

---

[۱] *Machine Learning*
[۲] *General Rule*
[۳] *Filter*
[۴] *Statistical*

به یکسری کلاس از پیش تعیین شده تعلق دارند انجام می‌پذیرد. در اینجا از یادگیری برای پیش بینی کردن[۱] کلاس نمونه‌ی داده‌ای جدیدی که به یکی از این کلاس‌های از پیش تعیین شده تعلق دارد، استفاده می‌شود. یک سیستم تشخیص نفوذ مبتنی بر طبقه بندی تلاش می‌کند که تمام ترافیک شبکه را به دو کلاس نرمال و نفوذ، و یا در سطح بالاتر به یک کلاس نرمال و چند کلاس نفوذ نسبت دهد. مسئله‌ی چالش‌زا در این روش تلاش برای کاستن *FN* (طبقه‌بندی رفتارهای نفوذی به عنوان نرمال) و *FP* (طبقه‌بندی رفتارهای نرمال به عنوان رفتار نفوذی) می‌باشد. در ادامه چند نمونه از اصلی‌ترین روش‌های طبقه‌بندی استفاده شده در سیستم-های تشخیص نفوذ مورد مطالعه اجمالی قرار گرفته است.

أ. تولید قوانین استنتاجی

تولید قوانین استنتاجی[۲]شامل روش‌هایی است که عموماً برای یادگیری قانون از روی نمونه‌ها استفاده شده‌اند و شامل روش درخت تصمیم[۳]و روش جداسازی و حل[۴]می‌باشند. در روش درخت تصمیم ابتدا یک درخت تصمیم ایجاد می‌شود و سپس این درخت به یک سری قانون تقسیم شده و در نهایت این قوانین ساده می-شوند. یکی از سیستم‌هایی که بر اساس این روش کار می‌کند *C4.5* است*[Aba02]*. در روش جداسازی و حل در هر مرحله یک قانون آموخته شده و سپس نمونه‌های پوشانده شده توسط آن قانون جدا می‌شوند و این روال بر روی نمونه‌های باقیمانده تکرار می‌شود. می‌توان سیستم *RIPPER* را پرکاربردترین نماینده برای تکنیک‌های جداسازی و حل برشمرد که در واقع یک برنامه یادگیری قانون می‌باشد*[Coh95]*.

از ویژگی‌های *RIPPER* می‌توان به سرعت بالا، دقت بالا و تولید قوانین کوتاه اشاره‌کرد. این روش بسیار با-ثبات[۵]بوده و بی‌شک یکی از بهترین الگوریتم‌های استفاده‌شده در آزمایشات و تحقیقات گذشته بوده است. این سیستم شامل مجموعه‌ای از قوانین شرکت‌پذیری و الگوهای پرتکرار[۶]بوده که می‌تواند برای طبقه‌بندی

---

[۱] *Prediction*

[۲] *Inductive*

[۳] *Decision tree*

[۴] *Separate and Conquer*

[۵] *Stable*

[۶] *Frequent Pattern*

ترافیک شبکه مورد استفاده قرار بگیرد. یکی از ویژگی‌های جالب این روش این است که قوانین تولید شده در آن بسیار ساده بوده و بررسی صحت این قوانین توسط افراد خبره به راحتی امکان پذیر می‌باشد.

در *[Lee99]* آقای *Lee* و همکارش سیستم *RIPPER* را در چارچوبی که از داده‌کاوی برای تشخیص نفوذ استفاده می‌کند، به کار گرفتند. این چارچوب شامل طبقه‌بندی، قوانین شرکت پذیری[۱]و الگوریتم‌های وقایع ضمنی[۲]بوده که می‌تواند به صورت اتوماتیک الگوهای تشخیص را تولید نماید. در این روش مشخص می‌شود که قوانین شرکت پذیری و وقایع ضمنی می‌توانند برای تخمین بهینه و محاسبه‌ی الگوهای استوار داده‌های بررسی مورد استفاده قرار بگیرند. از آنجا که اصلی‌ترین تکنیک داده‌کاوی به کار گرفته شده در این پایان‌نامه یک سیستم *RIPPER* می‌باشد، در پایان فصل به بررسی کامل این الگوریتم و دلایل انتخاب این روش پرداخته خواهد شد.

ب. الگوریتم‌های ژنتیک

الگوریتم ژنتیک یک روش حل مسائل بهینه‌سازی است که بر اساس فلسفه انتخاب اصلح در طبیعت[۳]ایجاد شده است و در حقیقت این فرایند از سیر تکامل زیستی ناشی می‌شود*[Gol89]*. الگوریتم ژنتیک با استفاده از تکرار و عملگرهای ژنتیکی، یک جمعیت[۴]را شامل مجموعه‌ای از افراد[۵]تغییر می‌دهد. سپس افرادی از جامعه که جواب‌های بهتری نسبت به دیگر افراد تولید می‌کنند از نسل جاری انتخاب کرده و از آنها برای تولید فرزندان در نسل‌های بعدی استفاده می‌کند. یک الگوریتم ژنتیک هنگامی متوقف می‌شود که شرط توقف اتفاق بیفتد. شرط توقف می‌تواند حداکثر تعداد نسل‌های تولید شده یا رسیدن به یک حد آستانه یا هر شرط دیگری باشد.

در جستجوی ژنتیک هر فرد از جامعه حاوی یک کروموزم است که این کروموزم به نوبه خود حاوی تعدادی ژن می‌باشد که خصوصیات آن فرد از جامعه را کد می‌کند. علاوه بر این هر الگوریتم ژنتیک دارای تابعی به نام

---

[۱] *Association Rules*

[۲] *Frequency Episode*

[۳] *Natural Selection*

[۴] *Population*

[۵] *Individual*

تابع شایستگی[۱]است که وظیفه آن بدست آوردن مقدار شایستگی می‌باشد. تابع شایستگی معیاری است که برای یک کروموزم ورودی که به ازاء یک فرد از جامعه به این تابع ارسال شده است بهینه می‌شود. به عبارت دیگر تابع شایستگی یک کروموزوم را به عنوان ورودی دریافت کرده سپس ژن‌های آنرا کدگشایی و بر این اساس عددی بدست می‌آورد که مقدار این عدد بیانگر میزان مناسب یا بهینه بودن این فرد از جامعه است.

سه عملگر ژنتیک معروف در زمینه الگوریتم ژنتیک عملگرهای جهش[۲]، تقاطع[۳]، انتخاب[۴] می‌باشند. عملگر جهش به طور تصادفی یا براساس یک الگوریتم خاص تعدادی از ژن‌های یک کروموزم یا یک فرد از جامعه که برای این عمل انتخاب شده است تغییر می‌دهد و فرد جدید را ایجاد می‌کند. در تقاطع دو کروموزوم از قسمت‌های مختلف شکسته می‌شوند و ژن‌های آنها با یکدیگر تعویض می‌شود و نمونه جدیدی ایجاد می‌گردد. این عملگر در حقیقت همانند ازدواج دو فرد با یکدیگر و تولد یک فرزند در نتیجه این ازدواج می‌باشد. عملگر انتخاب، والدین را برای ایجاد نسل بعدی بر اساس مقادیر خروجی تابع شایستگی برای آن‌ها، می‌یابد.

در *[Chi01]* از الگوریتم‌های ژنتیک و درخت تصمیم‌گیری برای نمایش داده‌ها استفاده شده است. در این تحقیق *Chittur* نرخ تشخیص بالا و نرخ هشدار غلط پایینی ارائه کرده است که به نوبه‌ی خود قابل توجه می‌باشد. در *[Cre95]* نیز *Crosbie* و همکارش از الگوریتم ژنتیک در درخت‌های کم‌پشت برای تشخیص رفتار غیرعادی استفاده کرده‌اند.

ج. منطق فازی

سال‌های اخیر گواهی بر گسترش سریع برنامه‌های کاربردی و سیستم‌های فازی از منظر تعداد و تنوع بوده است. در میان متدهای محاسبات نرم منطق فازی بیش از همه مورد توجه بوده و بیش از همه به کار گرفته شده است. ساختار اصلی یک سیستم استنتاج فازی شامل مدلی است که مشخصات یک ورودی را به توابع عضویت[۵]ورودی نگاشت می‌کند. انواع شناخته شده‌ای از سیستم‌های استنتاج فازی وجود دارند. مدل فازی

---

[۱] *Fitness Function*

[۲] *Mutation*

[۳] *Crossover*

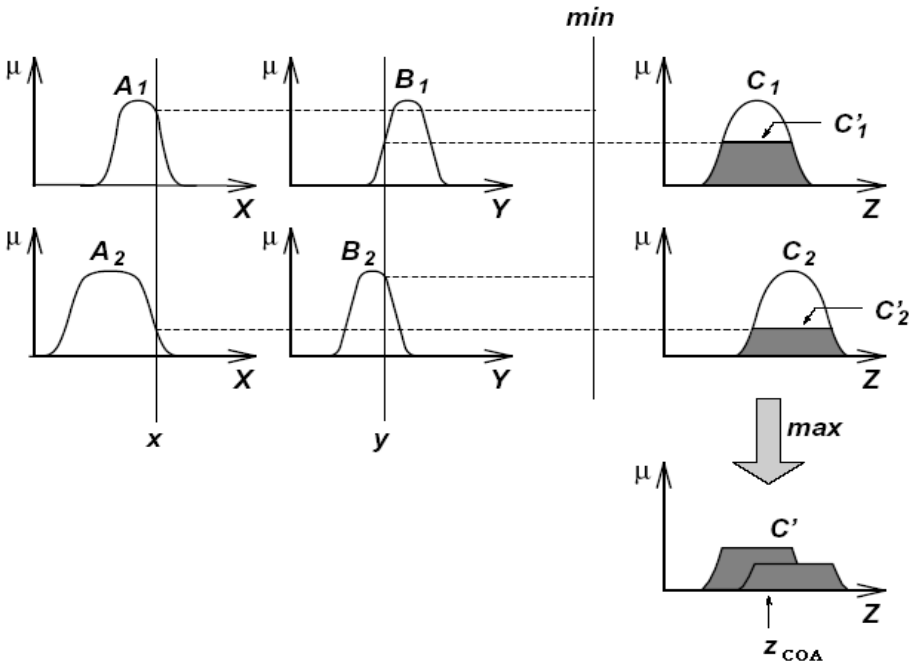
[۴] *Selection*

[۵] *Membership Functions*

ممدانی[۱] از اولین گام‌هایی بود که تلاش می‌کرد یک ورودی را بر اساس تجربه یک فرد خبره به یک فضای خروجی نگاشت کند. مثالی از مدل فازی ممدانی با دو قانون برای دو ورودی و یک خروجی در ادامه مشاهده می‌شود:

if x is A1 and y is B1 then Z is C1.
if x is A2 and y is B2 then Z is C2.

A و *B* مجوعه‌های فازی ورودی با توابع عضویت *A1* ، *A2* و *B1* ، *B2* هستند و *C* مجموعه فازی خروجی می‌باشد. ماکزیمم و مینیمم به ترتیب به عنوان عملگرهای *T- Norm* و *T-conorm* در نظر گرفته شده‌اند. سیستم استنتاج فازی حاصل در شکل ۳- ۲ نمایش داده شده است*[Jan93]*.



شکل ۳- ۲: سیستم استنتاج فازی ممدانی با دو ورودی و یک خروجی همراه با دو قانون و*min* و *max* به ترتیب به عنوان عملگر *T norm* و *T conorm*

از مزایای استفاده از سیستم‌های فازی در سیستم‌های کشف نفوذ می‌توان به این مورد اشاره کرد که بسیاری از انواع نفوذها به صورت قطعی[۲] قابل تعریف نبوده وهمچنین پیغام‌های هشدار نیز خود عموماً فازی هستند. علاوه بر این سیستم‌های فازی می‌توانند ورودی‌هایی که از منابع مختلف گردآوری شده را با یکدیگر ترکیب

[۱] *Mamdani Fuzzy Model*
[۲] *Crisp*

کنند*[Dik01]*. در *[Dik00]* مؤلفان داده‌ها را بر اساس معیارهای آماری مختلف طبقه‌بندی کرده، سپس قوانین منطق فازی را برای این داده‌ها تولید و بر اساس قوانین تولید شده آن‌ها را به دو کلاس نرمال و نفوذ طبقه‌بندی می‌کنند.

اکثر سیستم‌های فازی از یک فرد خبره برای ایجاد پایگاه داده‌ی قوانین فازی خود استفاده می‌کنند. اغلب در این سیستم‌ها فردی که آشنایی کامل با سیستم دارد، مجموعه‌ای از قوانین حسی خود را با استفاده از قوانین *if-then* فازی بیان می‌کند و سپس این قوانین در پایگاه دانش قوانین فازی قرار می‌گیرد و سیستم فعالیت-های خود را بر اساس این قوانین انجام می‌دهد. اما کسب دانش از متخصصین به دلایل متعددی سخت، همراه با اشتباه و یک پروسه وقت‌گیر و تکراری است. علاوه بر این سیستم‌های فازی معمولاً تطبیق‌پذیر نیستند بدین معنا که پس از آنکه قوانین فازی در پایگاه قوانین قرار داده شدند این قوانین تغییری نخواهند کرد و در صورتیکه سیستم با وضعیت‌های جدیدی رو به رو شود نه تنها قادر نیست قوانین جدید را به پایگاه قوانین اضافه کند، بلکه قادر به تغییر قوانین موجود نیز نمی‌باشد. بنابراین ساخت یک سیستم فازی با قابلیت‌های یادگیری و تطبیق پذیری بسیار مورد توجه قرار گرفته است*[Aba05]*.

د. شبکه‌های عصبی

شبکه عصبی مصنوعی یک سیستم پردازشی داده‌ها است که از مغز انسان ایده گرفته و پردازش داده‌ها را به عهده پردازنده‌های کوچک و بسیار زیادی سپرده که برای حل یک مسئله به صورت شبکه‌ای به هم پیوسته و موازی با یکدیگر رفتار می‌کنند. ایده اصلی آن در *1940* توسط *Walter Pitts* و *Waren Mc Culloch* با الگو گرفتن از عملکرد مدل نرون‌های عصبی مغز انسان مطرح شد. فرضیات مهم در شبکه‌های عصبی از این قرار است:

- داده پردازی اطلاعات در اجزای ساده به نام نرون[۱]صورت می‌گیرد.
- اطلاعات بین نرون‌ها از طریق ارتباطات آنها رد وبدل می‌شود.

[۱] *Neuron*

- هر یک از این رابطه‌ها دارای وزن W مختص خود هستندکه در مقدار اطلاعات رد وبدل شده با سایر نرون‌ها ضرب می‌شوند و به مرور زمان این وزن‌ها تنظیم می‌گردند. در واقع از این منظر است که شبکه از محیط تاثیر پذیرفته و آموزش می‌بیند.
- هر یک از نرون‌ها برای محاسبهٔ خروجی خود، دارای یک تابع فعالسازی[۱]است که معمولاً تابعی غیرخطی است و روی ورودی‌ها اعمال می‌شود.
- هر نرون در صورتی خروجی خواهد داشت که حاصل تابع فعالسازی آن از یک حد آستانه[۲]بیشتر شود.

امروزه شبکه‌های عصبی در کاربردهای مختلفی نظیر مسائل تشخیص الگو که خود شامل مسائلی مانند تشخیص نفوذ، تشخیص خط، شناسایی گفتار، پردازش تصویر و مسائلی از این دست می‌باشد استفاده می-شوند. آقای *McHugh* در *[McH00]* دلیل استفاده از متدهای تشخیص الگو و یادگیری از طریق نمونه‌ها در روش‌های تشخیص نفوذ را در دو گروه زیر دسته‌بندی کرده‌است:

- توانایی روش‌های یادگیری از طریق نمونه،[۳] امکان تشخیص نفوذهای جدید را برای سیستم فراهم می‌نماید.
- با کمک این روش‌ها امضای نفوذها به صورت اتوماتیک از روی داده‌های برچسب زده بدست می‌آید.

عقیده‌ی *Ghosh* بر این است که یک شبکه عصبی آموزش دیده‌ی قوی از نظر کارایی با سیستم‌های تطابق امضا اولیه قابل مقایسه بوده و در برخی مواقع عملکرد بهتری نیز خواهد داشت*[Gho99]*.

ه. ماشین بردار پشتیبان

مفهوم ماشین بردار پشتیبان[۴]توسط *Vapnik* ارائه شده است و به خانواده کلاس‌بندهای خطی تعمیم‌یافته تعلق دارد*[Vap95]*. درحالی‌که الگوریتم‌های یادگیری کلاسیک، کمینه کردن خطاها را به‌شکل تجربی انجام

---

[۱] *Activation Function*

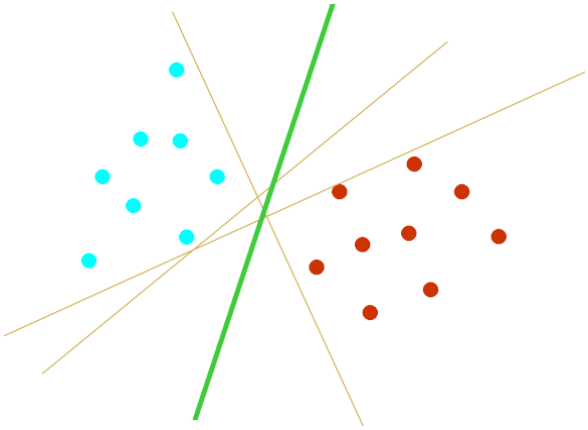
[۲] *Threshold*

[۳] *Learning by Example*

[۴] *Support Vector Machine (SVM)*

می‌دهند، *SVM* به‌شکل ساختاری به‌دنبال کمینه کردن احتمال دسته‌بندی نادرست داده‌ها می‌باشد. ایده‌ی اصلی به‌کارگرفته شده در *SVM* تلاش برای یافتن ابرصفحه جداکننده[۱] با حداکثر حاشیه[۲] جهت جداسازی نمونه‌های مثبت و منفی از یکدیگر می‌باشد *[Vap95]*.

در *[Muk03]*، *Mukkamala* و همکارانش از یک روش *SVM* برای تشخیص نفوذ استفاده کردند. آن‌ها از پنج *SVM* معمولی برای طبقه‌بندی اتصالات نرمال و چهار نوع نفوذ اصلی *DoS*، *U2R*، *R2L* و *Probe* موجود در داده‌های آزمایشی *KDD Cup* استفاده کرده‌اند. هر کدام از *SVM*ها کارایی‌ای در حدود *99%* داشته و آنها معتقدند که درمقایسه با بالاترین کارایی که با روش شبکه عصبی حاصل شده‌است *(87.07%)*، *SVM* از هر دو جنبه‌ی دقت و سرعت از شبکه‌های عصبی قوی‌تر می‌باشد.



شکل ۳- ۳: بهترین ابرصفحه برای جداسازی داده‌ها

۳-۳-۲-۲- تکنیک‌های خوشه‌بندی

خوشه‌بندی[۳] یکی‌از تکنیک‌های یادگیری ماشین توصیفی[۴] است که به دنبال پیدا کردن مجموعه متناهی از گروه‌ها برای توضیح دادن داده‌ها می‌باشد. اشیاء داخل یک گروه شباهت بیشتری با اشیاء بین گروهی دارند. در این نوع روش‌های یادگیری ماشین نیازی به تعریف کلاس‌های از پیش تعیین شده نبوده و کلاس‌ها از روی

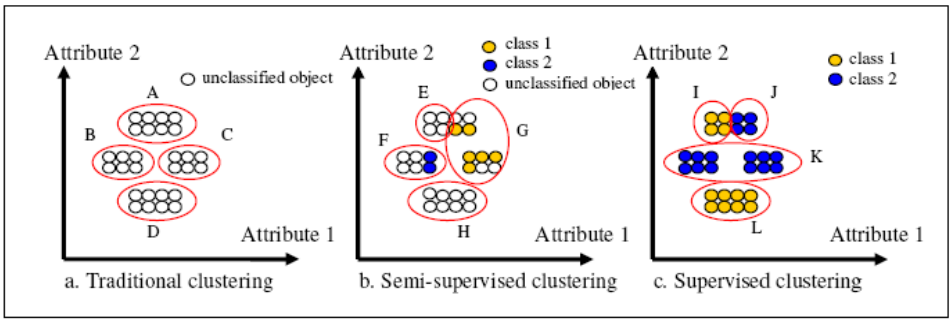
[۱] *Separating Hyper-plane*
[۲] *Maximum Margin*
[۳] *Clustering*
[۴] *Description*

داده‌ها استخراج می‌شوند. در واقع در خوشه‌بندی داده‌های آزمایش مشخص می‌کنند که چه چیز را باید یاد گرفت. تکنیک‌های خوشه بندی را می‌توان به سه گروه تقسیم‌بندی نمود.

۱- **خوشه‌های بدون نظارت**[۱]: در این حالت خوشه‌ها هیچ گونه اطلاعی از داده‌ها(برچسب) ندارند و بر اساس شباهت بین اشیاء داده‌ها را به یکسری خوشه تقسیم‌بندی می‌کنند.

۲- **خوشه‌های نیمه نظارت**[۲]: در این حالت نه تنها از شباهت بین داده‌ها بلکه از اطلاعات برچسب یکسری از داده‌ها نیز برای گروه‌بندی استفاده می‌شود. که در این حالت بایستی نمونه‌های با یک برچسب به یک خوشه تعلق داشته باشند و نمونه‌های با برچسب‌های مختلف متعلق به خوشه‌های مختلف هستند.

۳- **خوشه‌های نظارتی**[۳]: این الگوریتم بر روی داده‌های برچسب‌گذاری شده عمل کرده و برای داده‌های موجود در یک برچسب مشخص از عمل خوشه‌بندی استفاده می‌کند.



شکل ۳- ۴: انواع خوشه‌بندی

عموماً سیستم‌های تشخیص نفوذ سوءاستفاده نمونه‌ای از طبقه‌بندی می‌باشند که از طریق یکسری داده‌ی آزمایشی برچسب خورده مدل سیستم را آموخته و از آن برای پیش‌بینی کلاس نمونه‌های آتی استفاده می‌کند. سیستم‌های تشخیص نفوذ رفتار غیرعادی یک نمونه از حالت خوشه‌بندی بدون نظارت می‌باشند که از خوشه‌بندی برای پیاده‌سازی رفتار نرمال استفاده کرده و هرگونه انحراف از آن‌را بعنوان نفوذ معرفی می‌کنند.

---

[۱] *Unsupervised Clustering*

[۲] *Semi-Supervised Clustering*

[۳] *Supervised Clustering*

در این تحقیق برای بدست آوردن مدل مربوط به رفتارهای نفوذی از روش‌های طبقه‌بندی و برای بدست آوردن مدل مربوط به رفتارهای نرمال شبکه از یک تکنیک خوشه‌بندی استفاده نموده‌ایم.

روش *k-means* یکی از ساده‌ترین روش‌های خوشه‌بندی مورد استفاده در تشخیص نفوذ محسوب می‌شود. در این روش فرض بر این است که داده‌ها از چندین خوشه تشکیل شده‌اند و هر خوشه توسط یک مرکز خوشه مشخص می‌شود. در این روش مراکز خوشه‌ها با استفاده از کمینه کردن تابع خطا به‌صورت بهینه مشخص می‌شود. در *[Blo01]*، *Bloedron* از این روش برای خوشه‌بندی مجموعه وقایع ثبت شده‌ی اتصالات برای تشخیص نفوذ استفاده کرده است.

در *[Sta02]* نیز از شبیه‌سازی تبرید[۱]برای خوشه‌بندی رخدادها (بسته‌های غیرعادی) استفاده شده است. در این روش اتصالات هماهنگ[۲]اسکن پورت‌ها در یک خوشه قرار می‌گیرند. با این کار زمان اجرا از چند جمله‌ای به خطی تبدیل می‌شود.

۳-۳-۳- تکنیک‌های آماری

تکنیک‌های آماری[۳]از قدیمی‌ترین روش‌های بکار رفته جهت تشخیص نفوذ در شبکه‌های کامپیوتری می‌باشند. در این روش‌ها فرض بر این است که نمونه‌های غیرنرمال، ساختارهای متفاوتی نسبت به نمونه‌های نرمال دارند. بر این اساس می‌توان نمونه‌های نرمال را به کمک روش‌های آماری مدل‌سازی نمود. *NIDES/STAT* و *Haystack* گونه‌هایی از سیستم‌های آماری تشخیص رفتارغیر عادی به شمار می‌روند. به‌عنوان مثال در *NIDES/STAT* با استفاده از یک تابع ساده میزان انحراف نمونه از رفتارهای نرمال بدست می‌آید و از این طریق می‌توان ماهیت نمونه را شناسایی نمود. در این روش T^2 به‌عنوان میزان غیرطبیعی بودن نمونه معرفی می‌شود. چنانچه n معیار $m_1,m_2,\ldots,m_n$ برای مدل کردن هر نمونه مورد استفاده قرار گیرد و $S_1,S_2,\ldots,S_n$ نیز میزان انحراف در هر یک از این معیارها را نشان دهد، با فرض مستقل بودن معیارها از یکدیگر داریم:

---

[۱] *Simulated Annealing*

[۲] *Coordinated*

[۳] *Statistical Techniques*

$T^2 = {S_1}^2 + {S_2}^2 + \cdots + {S_n}^2.$

البته بایستی در بازه‌های زمانی مشخص معیارهای نرمال مورد بازنگری قرار گیرند تا کارایی سیستم افزایش یابد. *Haystack* نیز به‌عنوان یکی دیگر از روش‌های آماری تشخیص رفتار غیرعادی شناخته می‌شود. در این روش از الگوریتم آماری متفاوتی استفاده می‌شود که در مقایسه با *NIDES/STAT* توانایی تشخیص برخی از حملات شناخته شده نیز به آن اضافه شده است.

۳– ۴– تکنیک داده‌کاوی استفاده شده در این تحقیق

تکنیک‌های داده‌کاوی زیادی وجود دارد که امکان استخراج قوانین را از داده‌های برچسب خورده فراهم می‌نمایند. در این تحقیق برای آموزش قوانین از روش *RIPPER* استفاده شده است. در ادامه ابتدا جزئیات این روش بررسی شده و سپس دلیل انتخاب این روش بیان می‌شود. قبلاً در این فصل در مورد الگوریتم *RIPPER*[۱] یا همان *IREP** که در واقع نسخه‌ی بهبود یافته‌ی *IREP*[۲] می‌باشد صحبت شد ولی جزئیات این روش بیان نشد. در اینجا برای تشریح الگوریتم *RIPPER* می‌بایست ابتدا *IREP* تشریح شده و سپس جزئیات نسخه‌ی اصلاح شده‌ی آن در قالب *RIPPER* بیان شود. لازم به ذکر است که *IREP* نیز خود نسخه‌ی بهبود یافته‌ی *REP* است *[Coh95]*.

۳– ۴– ۱– الگوریتم *REP*

روش کار این الگوریتم بدین صورت بوده که ابتدا داده‌های آموزش به دو دسته‌ی آموزش اولیه *(70%)* و آموزش مجدد *(30%)* تقسیم می‌شوند. در فاز اول مجموعه‌ی قوانینی که فاقد کوچکترین خطا در مجموعه‌ی آموزش اولیه هستند تولید می‌شود. فاز دوم از این الگوریتم ساده سازی می‌باشد که یک استراتژی حریصانه[۳] برای ساده‌سازی مجموعه‌ی قوانین تولیدشده در فاز اول بکار گرفته می‌شود. در این فاز ممکن است یک قانون بطور کامل از مجموعه‌ی قوانین حذف شده و یا ترکیب ممکنی از شرط‌ها در یک قانون کاهش یابد (حذف شوند). این ساده‌سازی تا جایی ادامه می‌یابد که کمترین خطا در داده‌های آموزش مجدد وجود داشته باشد.

---

[۱] *Repeated Incremental Pruning to Produce Error Reduction*
[۲] *Incremental Reduced Error Pruning*
[۳] *Greedy*

۳-۴-۲- الگوریتم *IREP*

الگوریتم *IREP* یک روش تقسیم و غلبه می‌باشد و روش کار بدین صورت است که ابتدا یک قانون (با ارزش-ترین قانون) تولید شده و سپس داده‌هایی که این قانون می‌پوشاند از مجموعه‌ی داده‌ها حذف شده و داده-های باقی‌مانده مد نظر قرار گرفته و این کار تکرار می‌شود.

جزئیات این الگوریتم بدین صورت است که ابتدا داده‌های آموزش به دو بخش رویش[۱] *(70%)* و هرس[۲] *(30%)* تقسیم شده، و با ارزش‌ترین قانون در مجموعه‌ی رویش جستجو می‌شود. سپس این قانون با استفاده از مجموعه‌ی هرس، هرس می‌شود. پس از افزودن این قانون جدید به مجموعه قوانین، تمام داده‌های پوشانده شده از دو مجموعه‌ی رویش و هرس حذف می‌شوند. سپس مجموعه‌ی باقی مانده مانند قبل مجدداً به دو بخش رویش و هرس تقسیم شده و تمام مراحل بالا تکرار می‌شود. این کار تا زمانی که نمونه‌ی مثبتی برای پوشانده شدن وجود داشته باشد ادامه می‌یابد (شکل ۳- ۵).

```
procedure IREP(Pos, Neg)
begin
    Ruleset := ∅
    while Pos ≠ ∅ do
        *grow and prune a new rule*
        split(Pos,Neg) into (GrowPos, GrowNeg) and (PrunePos, PruneNeg)
        *GrowRule - propositional version of FOIL*
        Rule = GrowRule(GrowPos, GrowNeg)
        *prune the rule immediately*
        Rule = PruneRule(Rule, PrunePos, PruneNeg)
         if the error rate of Rule on (PrunePos, PruneNeg) exceeds 50% then
            return Ruleset
        else
            add Rule to Ruleset
            remove examples covered by Rule from (Pos, Neg)
        endif
    endwhile
    return Ruleset
end
```

شکل ۳- ۵: الگوریتم *IREP*

---

[۱] *Growing*

[۲] *Pruning*

در این روش برای هرس کردن یک قانون با استفاده از مجموعه‌ی هرس ممکن است هر ترتیبی از شرط‌ها از این قانون کاهش یابد (حذف شوند). در این حالت بایستی قانون به گونه‌ای تبدیل شود که رابطه زیر بیشترین مقدار ممکن شود:

$$v(Rule, PrunePos, PruneNeg) = \frac{p + (N - n)}{P + N}$$

در این حالت *p(n)* را تعداد نمونه‌های مثبت (منفی) پوشانده شده توسط این قانون از تعداد کل نمونه‌های موجود *P(N)* مثبت (منفی) در مجموعه‌ی هرس می‌باشد. به عبارت ساده‌تر قانون تا زمانی هرس می‌شود که مجموع تعداد نمونه‌های مثبت پوشانده شده *(p)* و منفی پوشانده نشده $(N - n)$ توسط این قانون بیشترین مقدار شود.

۳-۴-۳ الگوریتم *RIPPER*

همانطور که بیان شد *RIPPER* در واقع نسخه‌ی بهبود یافته‌ی *IREP* بوده و روش کار آن‌ها نیز بسیار شبیه به هم می‌باشند و تنها شروط خاتمه‌ی فاز رویش و فاز هرس تغییر می‌کنند. در فاز رویش شرط خاتمه‌ی جدیدی بر مبنای کمترین طول توصیفات[۱] بیان شده است. بدین‌صورت که در زمان تولید یک قانون جدید طول توصیفات جدید (D_{new}) نیز که تابعی از طول قانون و نمونه‌های پوشانده شده می‌باشد، تولید می‌شود. فاز رویش تا زمانی که داده‌های مثبت پوشانده شده وجود داشته و طول توصیف جدید نیز در شرط زیر صدق کند، ادامه می‌یابد.

$$D_{new} < d(bits) + min\, D_i$$

در شرط خاتمه‌ی فاز هرس *IREP* نیز تغییر کوچکی مشاهده می‌شود و به شرط زیر تبدیل می‌شود.

$$v^*(Rule, PrunePos, PruneNeg) = \frac{p - n}{p + n}$$

فرض کنید در قانون R_1 تعداد نمونه‌های مثبت پوشانده‌شده (p_1) برابر با ۲۰۰۰ و تعداد نمونه‌های منفی پوشانده‌شده (n_1) برابر با ۱۰۰۰ باشد و در قانون R_2 تعداد نمونه‌های مثبت پوشانده‌شده (p_2) برابر با ۱۰۰۰ و تعداد نمونه‌های منفی پوشانده‌شده (n_2) برابر با ۱ باشد:

R1: p1 = 2000, n1 = 1000

---

[۱] *Minimum Description Length (MDL)*

R2: p1 = 1000, n1 = 1

در این حالت در روش *IREP* قانون R_1 به عنوان قانون هرس شده انتخاب می‌شود که در واقع انتخاب نا-معقولی می‌باشد. ولی در روش *RIPPER* قانون R_2 به عنوان قانون هرس شده انتخاب می‌شود که در واقع روش معقولانه و کارامدتری است. به بیان ساده‌تر می‌توان گفت قانونی به عنوان قانون هرس شده انتخاب می-شود که تعداد نمونه‌های پوشانده شده‌ی مثبت آن نسبت به تعداد نمونه‌های پوشانده شده‌ی منفی آن در مقایسه با سایر قوانین بیشترین مقدار باشد.

الگوریتم دوکلاسه‌ی *RIPPER* به سادگی به یک الگوریتم چندکلاسه تبدیل می‌شود و روش انجام این کار بدین صورت است که ابتدا کلاس‌هایی که برچسب متفاوت دارند بر اساس تعداد تکرار نمونه‌های هر کلاس به صورت صعودی مرتب می‌شوند. یعنی اگر کلاس c_1 دارای کمترین تکرار و c_2 دارای تعداد تکرار بیشتر و... باشد در این حالت خواهیم داشت:

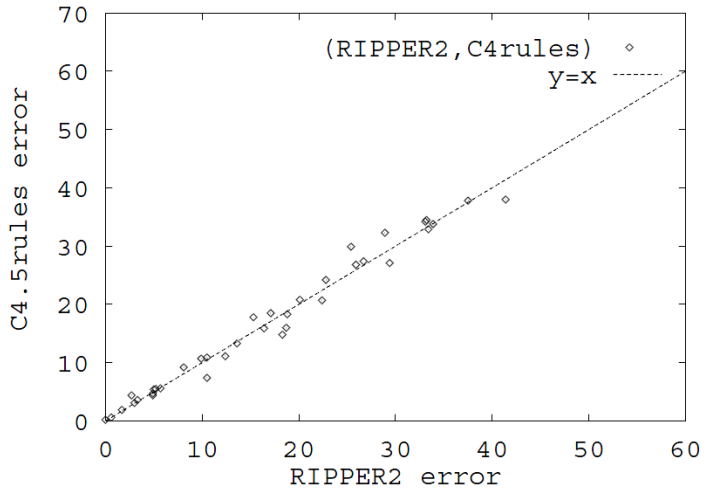
$c_1, c_2, c_3, \ldots$

در مرحله‌ی بعد *RIPPER* به دنبال مجموعه قوانینی می‌گردد که کلاس c_1 را از بقیه‌ی کلاس‌ها متمایز کند. این کار با فراخوانی الگوریتم *RIPPER* در حالتی که داده‌های مثبت آن دارای برچسب c_1 و داده‌های منفی آن دارای برچسب $c_2, c_3, \ldots$ می‌باشد انجام می‌شود. پس از این کار تمام نمونه‌های پوشانده شده توسط قوانین فراگرفته شده از مجموعه‌ی داده‌ها حذف شده و در مرحله‌ی بعد سعی می‌شود کلاس c_2 از بقیه‌ی کلاس‌های باقی مانده $(c_3, c_4, \ldots)$ جدا شده و قوانین مربوط به آن استخراج شود. این کار تا جایی تکرار می‌شود که تنها مجموعه‌ای که بیشتری فراوانی را دارد باقی بماند، در این حالت این کلاس به عنوان کلاس پیش فرض انتخاب می‌شود.

```
procedure RIPPER(Pos, Neg)
begin
    Ruleset := ∅
    Ruleset := IREP*(Pos, Neg)
    Repeat k times
        Ruleset := MDLOptimize(RuleSet)
    Pos* := positive examples not covered by RuleSet
    RuleSet := RuleSet + IREP*(Pos*, Neg)
    return Ruleset
end
```

شکل ۳- ۶: الگوریتم *RIPPER*

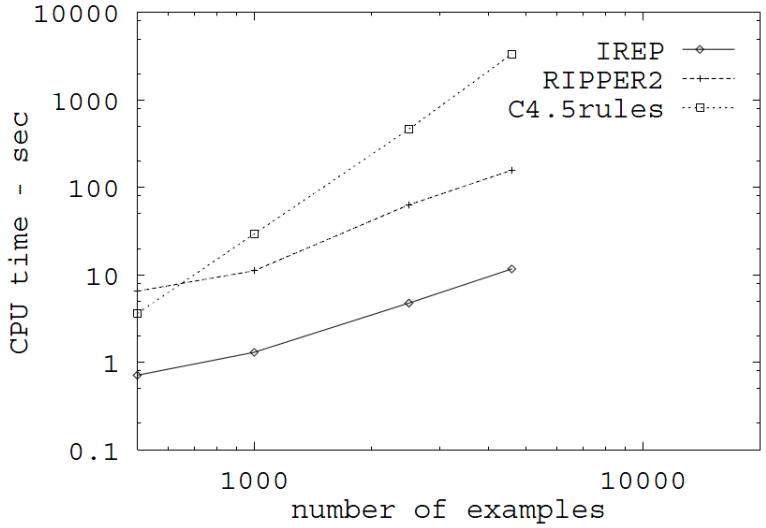
در ادامه اندکی درباره‌ی دلیل گزینش این الگوریتم به‌عنوان روش داده‌کاوی مورد استفاده در این تحقیق توضیح داده شده است. همانطور که می‌دانیم الگوریتم‌های فراوانی وجود دارند که در یادگیری قوانین از مجموعه‌ی داده‌های آزمایشی به‌کار گرفته می‌شوند. از آنجا که در این تحقیق هدف گنجاندن قوانین تولید شده در قالب ویژگی‌های انواع حملات مختلف در ساختار درختی آنتولوژی خود می‌باشد، لذا علاوه بر دقت بالای الگوریتم، تولید قوانین کم‌تعداد، کوتاه و قابل فهم نیز بسیار اهمیت دارد. از این رو به بررسی معروف-ترین این روش‌ها پرداخته پس از اجرای این الگوریتم‌ها بر روی یک مجموعه‌ی آزمایشی مشخص شد که *RIPPER* و *C4.5* بیشترین دقت را دارند. اگرچه الگوریتم *C4.5* در اغلب موارد دقیق‌تر بوده ولی تعداد قوانین زیادی تولید می‌کند که پیچیده و غیر قابل فهم می‌باشند.



شکل ۳- ۷ : مقایسه‌ی میزان خطای الگوریتم‌های *RIPPERk* و *C4.5 [Coh95]*

الگوریتم *RIPPERk* در واقع همان الگوریتم *RIPPER* می‌باشد با این تفاوت که مرحله‌ی بهینه‌سازی و هرس *k* مرتبه تکرار شده است. الگوریتم *RIPPERk* نسبت به *C4.5* دارای سرعت و دقت بیشتری می‌باشد و به داده‌های دارای نویز نیز حساسیت کمی نشان می‌دهد. با بررسی‌های بیشتر مشخص شد که *RIPPER* و *C4.5* جزء پرطرفدارترین روش‌ها در بین محققان نیز بوده و در مقالات مختلف به مقایسه‌ی این دو روش پرداخته شده است. در اینجا برای نمونه مقایسه‌های انجام شده بین دو الگوریتم *RIPPER2* و *C4.5* توسط آقای *Cohen* از منظر سرعت و دقت نشان داده شده است (شکل ۳- ۷ و شکل ۳- ۸) *[Coh95]*.

پس از انجام این بررسی‌ها بر آن شدیم تا در این تحقیق از *RIPPER10* استفاده شود که هم دارای دقت بسیار زیادی می‌باشد و هم قوانین محدود و ساده‌ای تولید می‌کند و از آنجا که فرایند تولید آنتولوژی فرایندی دستی است لذا این‌گونه قوانین برای هدف ما بسیار مناسب می‌باشند.



شکل ۳- ۸ : مقایسه‌ی سرعت الگوریتم‌های *RIPPERk* و *C4.5 [Coh95]*

۳- ۵- خلاصه

در این فصل تکنیک‌ها و روش‌های مختلف داده‌کاوی مورد بررسی قرار گرفت و نمونه‌هایی از کاربرد این تکنیک‌ها در سیستم‌های تشخیص نفوذ بیان شد. تکنیک‌های مختلف داده‌کاوی استفاده شده در تشخیص نفوذ را می‌توان در غالب انتخاب خصیصه‌ها، یادگیری ماشین، تکنیک‌های آماری و... دسته بندی کرد.

تکنیک‌های مختلف یادگیری ماشین پرکاربردترین روش‌های داده‌کاوی در سیستم‌های تشخیص نفوذ بوده و خود به دو نوع طبقه‌بندی و خوشه‌بندی تقسیم می‌شوند.

در روش‌های طبقه‌بندی عمل یادگیری از طریق داده‌های آزمایشی‌ای که به یکسری کلاس از پیش تعیین‌شده تعلق دارند بدست می‌آید. سپس از آن برای پیش‌بینی کردن[۱]کلاس نمونه داده‌های جدیدی که به یکی از این کلاس‌های از پیش تعیین شده تعلق دارند، استفاده می‌شود. در این حالت کلاس‌های از پیش تعیین شده تعیین می‌کنند که چه چیزی را باید یاد بگیریم. ولی در روش‌های خوشه‌بندی کلاس‌های از پیش تعیین شده نیاز نبوده و کلاس‌ها از روی داده‌ها استخراج می‌شوند. به عبارت دیگر در خوشه‌بندی داده‌های آزمایش مشخص می‌کنند که چه چیز را باید یاد گرفت.

تکنیک‌های طبقه‌بندی نیز خود شامل روش‌های مختلفی مانند تولید قوانین استنتاجی، الگوریتم‌های ژنتیک، منطق فازی، شبکه‌های عصبی و ماشین‌های بردار پشتیبان و... می‌باشند که به طور مختصر مورد بررسی قرار گرفتند. در انتهای فصل نیز به بررسی جزئی‌تر تکنیک داده‌کاوی استفاده شده در این تحقیق پرداخته شد.

[۱] *Prediction*

فصل چهارم

«وب معنایی و کاربرد آن در تشخیص نفوذ»

۴-۱- مقدمه

این فصل شامل دو بخش اصلی می‌باشد. در بخش اول فصل مروری بر وب‌معنایی و تعاریف آن آورده شده است و در ادامهٔ آن بحث وب‌معنایی در امنیت اطلاعات و تحقیقات انجام شده در زمینه‌ی سیستم‌های تشخیص نفوذ مورد بررسی قرار گرفته است. در بخش دوم نحوه‌ی طراحی آنتولوژی‌های مبتنی بر *OWL* و به طور کلی اجزا و ویژگی‌های این نوع آنتولوژی‌ها به تفصیل شرح داده خواهد شد.

۴-۲- مروری بر وب معنایی

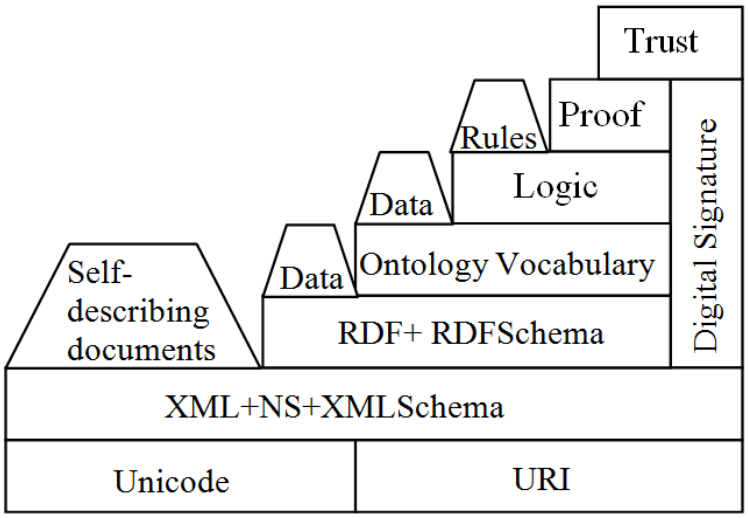
وب کنونی برای استفاده‌ی انسان‌ها ایجاد شده است، در این وب انسان‌ها بایستی بتوانند اطلاعات مورد نظر خود را مشاهده کرده، آن‌ها را فهمیده، اطلاعات مورد نظر را انتخاب و در صورت لزوم آن‌ها را جابجا کنند. اما هدف از طراحی وب معنایی این بوده که علاوه بر انسان ماشین نیز بتواند با وب کار کند و مفاهیم آن را درک کند، این امر نیازمند آن است که کدینگ (رمزگذاری) اطلاعات به گونه‌ای باشد که درک آن برای ماشین و برنامه‌های کامپیوتری راحت باشد. برای دستیابی به این روش کدکردن اطلاعات نیازمند بهره‌گیری از آنتولوژی‌ها می‌باشیم. با استفاده از آنتولوژی‌های قابل فهم برای ماشین، می‌توان وب را از حالتی که تنها برای انسان‌ها قابل فهم و استفاده است خارج کرده و آن را به سمت وب قابل فهم برای ماشین سوق داد.

در وب‌معنایی، داده خود قسمتی از وب است که می‌تواند مستقل از برنامه کاربردی، محیط و دامنه، پردازش شود. این مفهوم با اینترنت امروزه در تضاد است، زیرا اسناد پیدا شده توسط کامپیوتر باید جداگانه توسط فرد بازبینی شوند و اطلاعات مفید از آن استخراج شود. کامپیوترها اطلاعات را ارائه می‌دهند، ولی نمی‌توانند آن‌ها را بفهمند و داده‌های مربوط و مورد نیاز را نمایش دهند.

در پیاده‌سازی وب‌معنایی لازم است که ابرداده‌ی[۱] معنایی یا داده‌ای که داده را توصیف می‌کند، به منابع اطلاعاتی اضافه شود. در نتیجه ماشین‌ها می‌توانند داده‌ها را بر اساس اطلاعات مفهومی که آن‌ها را توصیف می‌کنند، پردازش کنند.

---

[۱] *Meta Data*



شکل ۴- ۱: لایه‌های تشکیل دهنده‌ی وب معنایی

اولین گام در ماشین‌ها جهت فهمیدن داده‌ها این است که داده را در فرمت یکسان دریافت کنند، مثلا یک فیلد که "خیابان" نامیده می‌شود، همه جا یک معنا و مفهوم داشته باشد. گام اصلی در وب معنایی این است، که داده‌ها از دامنه‌های مختلف، بر اساس ویژگی‌ها و رابطه‌هایشان با داده‌های دیگر کلاس‌بندی شوند. اینجاست که تکنولوژی‌های وب معنایی یعنی *RDF*، *RDFS* و *OWL*[۱] کاربرد پیدا می‌کنند.

۴-۲-۱- آنتولوژی

در مواردی که لازم است در یک حوزه‌ی خاص اطلاعات به اشتراک گذاشته شوند، به کمک آنتولوژی می‌توان به یک فرهنگ واژه مشترک دست پیدا کرد که در کل آن حوزه قابل استفاده و قابل فهم باشد. به کمک آنتولوژی می‌توان از مفاهیم اصلی در هر دامنه، تعاریفی قابل فهم برای ماشین ایجاد کرد و همچنین روابط فی‌مابین این مفاهیم را بیان کرد*[Nat01]*. در هوش مصنوعی نیز تعاریف مختلفی از آنتولوژی ارائه شده است:

- یک آنتولوژی توصیفی فرمال و صریح از ادراکات به اشتراک گذاشته شده است*[Bor97]* .
- یک آنتولوژی تعاریف واژگان معمول برای محققانی است که اطلاعات یک دامنه را به اشتراک می-گذارند*[Gru93]*.

---

[۱] *Ontology Web language* ، بر روی*RDF* و *RDFS* ساخته شده و معنای مشابه واژه‌ها در محیط‌های مختلف را تضمین می-کند. مثلاً با استفاده از*OWL* می‌توان گفت که "*person*" در واژگان *A* همان "*user*" در واژگان *B* است. یا منبع *A* و منبع *B* به یک چیز اشاره می‌کنند.

به اشتراک گذاشتن درک مشترک از ساختار اطلاعات، مابین انسان‌ها یا عامل‌های نرم‌افزاری، یکی از مهم‌ترین دلایل طراحی و توسعه‌ی آنتولوژی‌ها می‌باشد. برخی از دلایل علاقه‌مندی محققان به طراحی و ایجاد آنتولوژی را می‌توان به‌صورت زیر خلاصه نمود:

۱. به اشتراک گذاشتن ادراک و دریافت‌های رایج از ساختار اطلاعات مابین انسان‌ها و عامل‌های نرم-افزاری
۲. استفاده مجدد از دامنه‌ی دانش
۳. وضوح بخشیدن به فرضیات در نظر گرفته شده در یک دامنه
۴. جدا کردن دامنه‌ی دانش از دانش عملیاتی
۵. آنالیز کردن دامنه‌ی دانش

در تعریف مورد نظر ما، آنتولوژی عبارت است از: شرح واضح و فرمال از مفاهیم موجود در دامنه‌ی تحت بررسی، (غالباً مفاهیم را با کلاس‌ها نشان می‌دهند) و ویژگی‌های این مفاهیم، خصایص و صفات آنها. یک آنتولوژی به همراه نمونه‌های منحصر به فرد از کلاس‌هایش تشکیل پایگاه دانش می‌دهد.

۴– ۳– وب معنایی در امنیت اطلاعات

تکنیک‌ها و روش‌هایی که وب معنایی برای دستیابی به اهدافش استفاده می‌کند، مانند کمک گرفتن از مفهوم "محتوی و مضمون" می‌تواند در بسیاری از مباحث مطرح در علوم مهندسی کامپیوتر کاربرد داشته باشد. در این بخش کاربردهای وب‌معنایی و تکنیک‌هایش در مقوله‌ی امنیت اطلاعات مورد بررسی قرار گرفته است.

آقای *Victor Raskin* و همکارش در سال ۲۰۰۱ به بحث استفاده از آنتولوژی در امنیت اطلاعات پرداخته و دلایل مفید بودن استفاده از آنتولوژی را بررسی کرده‌اند*[Ras01]*. آن‌ها به این نتیجه رسیدند که با استفاده از آنتولوژی می‌توان به یک ابزار طبقه‌بندی قوی برای مجموعه‌ی نامحدودی از رویدادها دست پیدا کرد. هر یک از تکنیک‌ها و فن‌آوری‌های تأمین کننده‌ی امنیت اطلاعات که برای انجام کارها و وظایف خود بتوانند، از مفهوم "مضمون و محتوی" استفاده کنند، می‌توانند از روش‌های وب معنایی کمک بگیرند. در زمینه‌ی سیستم‌های تشخیص نفوذ تحقیقاتی انجام شده است، که در بخش بعدی به این مباحث بیشتر پرداخته و

پروژه‌ها و تحقیقاتی که در این زمینه انجام شده مورد بررسی قرار گرفته است. اما در سایر زمینه‌های مربوط به امنیت اطلاعات، اعم از زمینه‌های مربوط به فیلترینگ، دیواره آتش و... هنوز تحقیق و یا پروژهٔ خاصی انجام نشده، ولی با توجه به طریقه‌ی کارکرد و چگونگی انجام فعالیت‌هایشان می‌توان امیدوار بود که با استفاده از مفاهیم و تکنیک‌های وب معنایی می‌توان به این سیستم‌ها کمک کرد. به عنوان مثال یک سیستم فیلترینگ را در نظر بگیرید که قرار است مانع ورود اطلاعات خاصی به سیستم شود، در صورتی‌که سیستم فیلترینگ قادر باشد تا مطالب و محتوی سایت‌های ورودی را نیز مورد پردازش قرار دهد و آن‌ها را بررسی کند، قادر خواهد بودعمل فیلترینگ را به طور کامل‌تر و بهتری انجام دهد. البته هنوز در این زمینه تحقیقات خاصی انجام نشده است. تأکید این پایان‌نامه بر روی بحث استفاده از مفاهیم وب‌معنایی و مفهوم آنتولوژی در سیستم‌های تشخیص نفوذ می‌باشد.

۴– ۳– ۱– وب معنایی در سیستم‌های تشخیص نفوذ

همانطور که پیش‌تر نیز اشاره شد، وب‌معنایی به هدف توانمندسازی نظام‌ها برای درک محتوای اسناد و مدارک وب و برقراری ارتباط میان این اسناد و مدارک از طریق محتوا اشاره دارد و به کمک آن نظام‌ها قادر به تفسیر معانی هر سند و مدرک خواهد شد. در این حالت نه تنها محتوای منابع به صورت صحیح بیان می‌شوند، بلکه استدلال‌ها و دانش جدید نیز می‌تواند کشف شود. با استفاده از این دانش جدید وب می‌تواند سرویس‌دهی و کارایی بهتری داشته باشد. با توجه به این ویژگی وب‌معنایی واضح است که استفاده از تکنیک‌ها و روش‌های آن می‌تواند کمک شایانی در شناسایی و کشف نفوذها و رفتارهای مشکوک بنماید.

استفاده از مفهوم، مضمون و محتوی برای تشخیص و کشف رفتارهای مشکوک و تهدید کننده و همچنین شناسایی حملات شناخت شده به سیستم‌های کشف نفوذ بسیار کمک می‌کند. با استفاده از مفهوم آنتولوژی می‌توان به سیستم کشف نفوذ این امکان را داد که با آنالیز و بررسی اطلاعات قبلی به اطلاعات و داده‌های جدید و مفیدی دست پیدا کند. این روش به تشخیص رفتارهای مشکوک و شناسایی رفتارهایی که ممکن است حمله و یا نفوذ باشند، بسیار کمک می‌کند. یک سیستم تشخیص نفوذ برای تشخیص نفوذهای احتمالی به سیستم تحت نظارتش کارهای زیر را انجام می‌دهد:

۱. نظارت و آنالیز کردن کاربران و فعالیت‌های سیستم.

۲. بازرسی سیستم و پیکره‌بندی نفوذپذیری‌ها.

۳. ارزیابی صحت و درستی فایل‌ها و داده‌های حیاتی سیستم.

۴. شناسایی الگوهایی که حملات شناخته شده را منعکس می‌کنند.

۵. تجزیه و تحلیل‌های آماری برای فعالیت‌های غیر نرمال.

۶. پیگیری داده‌ها از نقطه ورود تا خروج.

با افزودن روش‌ها و مفاهیم وب‌معنایی می‌توان هر یک از موارد فوق را بهبود بخشید و روند انجام آنها را آسان‌تر و ساده‌تر کرد. مثلاً شناسایی حملات شناخته شده با استفاده از این روش‌ها بهتر و دقیق‌تر انجام خواهد گرفت. می‌توان از این روش‌ها به طور گسترده در سیستم‌هایی که بر مبنای تشخیص رفتارهای مشکوک[۱] هستند، استفاده کرد. با توجه به اطلاعاتی که در رابطه با حملات و نفوذهای شناخته شده در سیستم وجود دارد و وقایع رخ داده در سیستم، رفتارهای مشکوکی که ممکن است نفوذ باشند شناسایی می‌شوند. درواقع می‌توان گفت، که با استفاده از مفاهیم وب‌معنایی و روش‌های آنتولوژی تشخیص رفتارهای غیرطبیعی و سوء استفاده‌ها آسان‌تر و دقیق‌تر خواهد بود. تمامی این‌ها سبب شده تا کارایی و بهره‌وری سیستم تشخیص نفوذ افزایش پیدا کند و یا به عبارت بهتر میزان عدم تشخیص حمله و هشدار غلط این سیستم‌ها کاهش یابد.

۴– ۳– ۲– مرور تحقیقات انجام شده

استفاده از مفاهیم وب‌معنایی در سیستم‌های کشف نفوذ مبحث جدیدی است و پروژه‌ها و تحقیقات اندکی در این زمینه انجام شده است. تاکنون آنتولوژی در طراحی سیستم‌های تشخیص نفوذ پنج کاربرد داشته است:

۱. استفاده از آنتولوژی در تشخیص حملات و نفوذهای توزیع‌شده که این آنتولوژی نشان دهنده‌ی مدلی از حملات کامپیوتری می‌باشد *[Und03][Und04]* .

۲. استفاده از آنتولوژی به عنوان یک لایه‌ی دانش مشترک بین تمام عامل‌ها در یک سیستم تشخیص نفوذ *Outbound* که باعث ساده‌تر شدن تعاملات عامل‌های مختلف با هم می‌شود *[Man05]*.

---

[۱] *Suspicious*

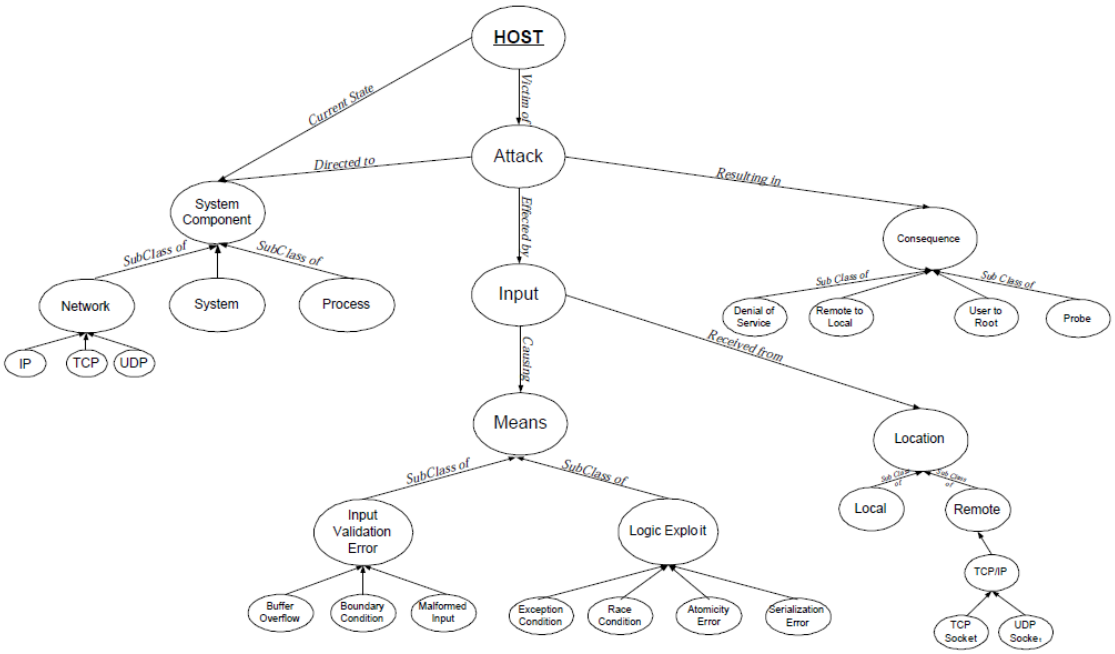
۳. استفاده از آنتولوژی در یک سیستم تشخیص نفوذ همکاری کننده که نشان‌دهنده‌ی ویژگی‌های قابل مشاهده در سنسورهای مختلف سیستم می‌باشد *[Yan04]* .

۴. استفاده از آنتولوژی برای نشان دادن اثر حمله‌ها (*ASO*) *[Ana05]*.

۵. استفاده از آنتولوژی در یک سیستم تشخیص نفوذ توزیع شده جهت شناسایی حملات کلاس *DoS* *[Abd07][Abd09]*.

در ادامه هر یک از تحقیقات معرفی شده به‌طور خلاصه مورد بررسی قرار گرفته است.

اولین پروژه‌ای که در این راستا مطرح شد در سال ۲۰۰۳ در دانشگاه مریلند به انجام رسید*[Und03]*. در این پروژه از یک آنتولوژی برای تعیین مدلی برای حملات کامپیوتری استفاده شده است. مبنای این آنتولوژی بر پایه‌ی آنالیز بیش از ۴۰۰۰ کلاس از نفوذها و استراتژی‌های حمله‌ی متناظر آن‌ها می‌باشد، که با توجه به اجزای سیستم هدف، اهداف حملات، نتایج حملات و محل حمله کننده، دسته‌بندی می‌شوند.. مدل استفاده شده در این تحقیق ، یک آنتولوژی *target_ centric* می‌باشد. آنتولوژی مورد استفاده در شکل ۴- ۲ نشان داده شده است.



شکل ۴- ۲: آنتولوژی *target_ centric*

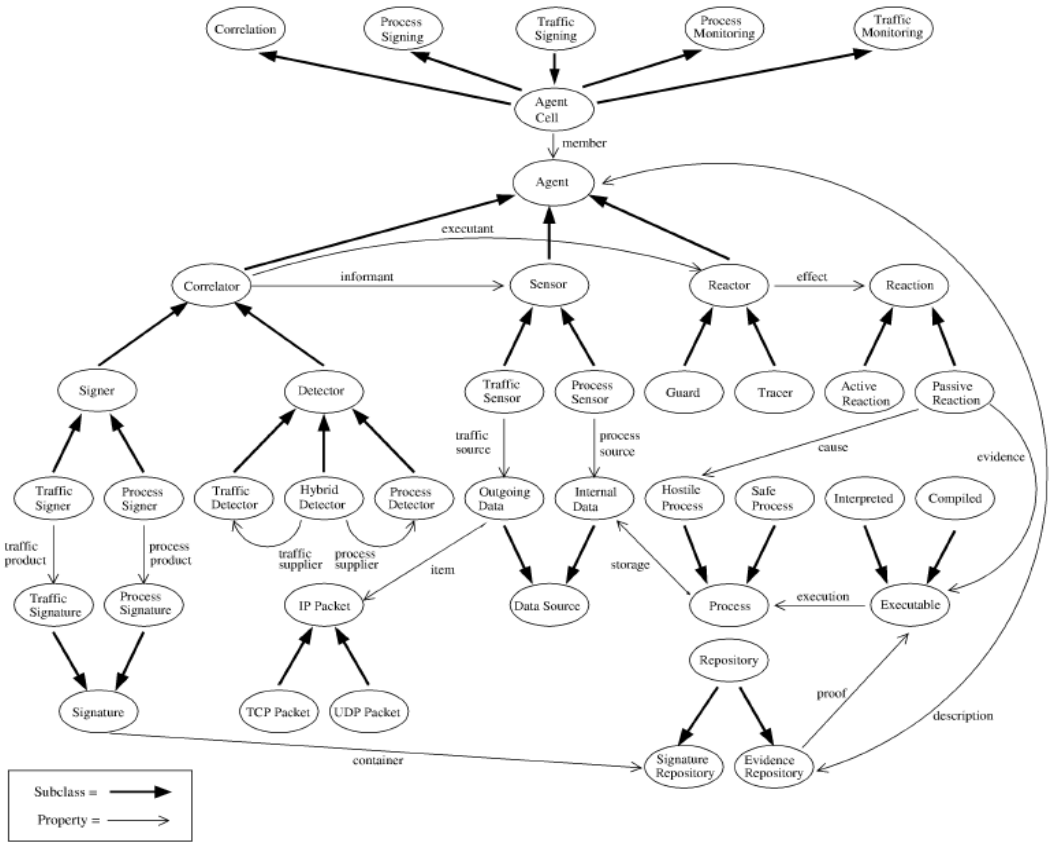
در این تحقیق یک مدل داده‌ای به عنوان آنتولوژیی که دامنه‌ی حملات کامپیوتری و نفوذها را مشخص می-کند، ساخته شده است. برخلاف تاکسونومی، آنتولوژی‌ها دارای ساختارهای قوی بوده که این ساختارها شامل تعاریفی از مفاهیم درون دامنه و رابطه‌ی بین آنها بوده که این تعاریف توسط ماشین قابل تفسیرپذیرمی‌باشند. بنابراین آنتولوژی‌ها سیستم نرم‌افزاری را قادر می‌سازند تا درک و برداشت مشترکی از اطلاعات داشته باشند و همچنین توانایی بیشتری برای استدلال و آنالیز این اطلاعات به سیستم نرم‌افزاری می‌دهند. درواقع هدف اصلی طراحی آنتولوژی‌ها به اشتراک‌گذاری دانش و استفاده‌ی مجدد بین موجودیت‌ها در یک دامنه می‌باشد. در اینجا سیستم‌های کشف نفوذ، سنسورهای شبکه و میزبان‌ها در واقع همان موجودیت‌های درون دامنه می-باشند.

از دیگر موارد استفاده از آنتولوژی در طراحی سیستم‌های کشف نفوذ می‌توان به پروژه‌ای که در سال ۲۰۰۵ در مکزیک انجام شده اشاره کرد*[Man05]*. در این پروژه از یک معماری چندعاملی مبتنی بر آنتولوژی در یک سیستم تشخیص نفوذ *outbound* استفاده شده است. روش کار یک روش مانیتورینگ بوده و سعی دارد مانع سوء استفادهٔ حمله‌کننده‌ها از سیستم‌های محلی برای مصالحه و توافق با سایر کامپیوترها شود.

در این پروژه استفاده از آنتولوژی در معماری سیستم کشف نفوذ سبب هوشمندتر شدن رفتار عامل‌ها شده و موجب می‌شود عامل‌ها ارتباطات و تقابلات‌شان را بهینه کرده و ارتباطات متقابل اجزای معماری قانون‌مند شود. در اینجا با بهره‌گیری از آنتولوژی مدل داده‌ای که نفوذ را نشان می‌دهد از منطق سیستم کشف نفوذ جدا می‌شود. آنتولوژی مورد استفاده در این پروژه بسطی از مدل *target-centric* می‌باشد، با این تفاوت که این آنتولوژی مبتنی بر حمله‌کننده می‌باشد. آنتولوژی مورد استفاده این اجازه را به اجزای معماری می‌دهد تا به طور مشابه المان‌هایی را که در محیط شرکت دارند تفسیر کنند. این کاهش در مضمون ارتباطات سبب ساده شدن پروتکل‌های ارتباطی شده و اینترفیسی ایجاد می‌کند که سایر عامل‌ها و نرم‌افزارهای کاربردی برای دستیابی به معماری تشخیص نفوذ *OID* از آن استفاده می‌کنند.

در واقع معماری *OID* مبتنی بر آنتولوژی، عامل‌ها را تحت محیط‌های اجرایی که خانه‌های عامل نامیده می-شوند سازماندهی می‌کند. آنتولوژی *attacker-centric* به عنوان یک لایه‌ی دانش مشترک برای تمام عامل‌ها

عمل می‌کند و نمایش دهنده‌ی عامل‌های ساکن در محیط بوده و تمام پیام‌های مکاتبات عامل‌ها[۱]برمبنای آنتولوژی می‌باشد و همین امر باعث کاهش سایز پیام‌ها شده است. در این پروژه آنتولوژی به یک ابزار تشخیص نفوذ *Outbound* که با نرم‌افزار مبتنی بر عامل ساخته شده است افزوده شده و این کار باعث افزایش کارایی شده است. این افزایش کارایی به دلیل کاهش سایز پیام‌ها و تسریع تفسیر دانش با استفاده از ساختار *inference-support* که در *RAM* عامل‌ها قرار می‌گیرد(یعنی کاهش پردازش لغوی و افزایش قابلیت استنتاج) می‌باشد. آنتولوژی مورد استفاده در این سیستم در شکل ۴- ۳ نشان داده شده است.



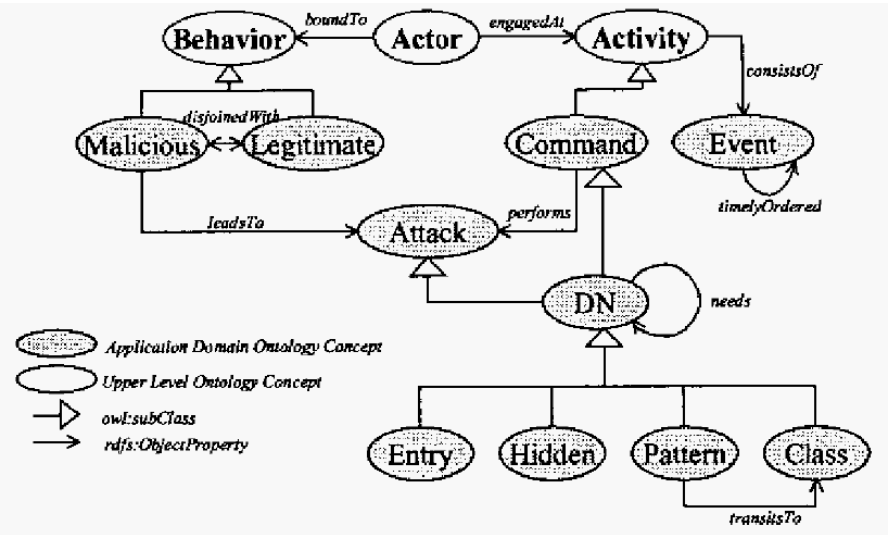
شکل ۴- ۳: آنتولوژی *attacker-centric*

کار دیگری که در این زمینه انجام شده توسط آقای *Wei Yan* و همکارانش در سال ۲۰۰۵ می‌باشد*[Yan05]*. ایشان از طرح جدیدی که از ترکیب پردازش زبان طبیعی (*NLP*) و وب‌معنایی ایجاد شده است، برای بازیابی و استخراج سناریوی حمله‌ها استفاده کرده‌اند. با استفاده از اصول، قوانین و متدهایی که در *NLP* وجود دارد،

[۱]*Agent Communicated*

اطلاعات معنایی از هشدارهای سیستم تشخیص نفوذ استخراج می‌شود. طرح و الگوی معنایی مطرح شده در اینجا فرمت ترکیبی هشدارها را به جریان هشدار معنایی قابل فهم برای ماشین تبدیل می‌کند، این کار با استفاده از گرامر *PCTCG*[۱] و آنتولوژی تعریف شده در دامنه‌ی امنیت نفوذ انجام می‌گیرد، که برای یکپارچه-سازی هشدارهای خام دریافت شده از سنسورهای سیستم‌های تشخیص نفوذ ناهمگون، بسیار مفید می‌باشد. در این روش آنتولوژی، معنایی که برای دامنه‌ی امنیت سیستم تشخیص نفوذ است بر مبنای سؤالاتی که مدیر امنیت به طور طبیعی می‌پرسد (کی و کجا و با چه قصدی عمل رخ داده؟ و چه نتیجه‌ای گرفته شده-است؟)، ساخته می‌شود.

یکی دیگر از تحقیقات صورت گرفته در این زمینه در سال ۲۰۰۵ در دانشگاه آتن و توسط *AnagnostopouIos* و همکارانش انجام شده است*[Ana05]*، در این پروژه تحقیقاتی به مسألهٔ پیشگویی حملات و کلاس‌بندی اهداف حمله‌کننده‌ها توجه شده است. در این روش پیشنهادی از یک کلاس‌بند *Bayesian* و یک الگوریتم استنتاج احتمالی استفاده می‌شود. الگوریتم استنتاج با بکارگیری اطلاعاتی که سیستم تشخیص نفوذ ترکیبی با استفاده از متد آنتولوژی ایجاد کرده است، کار خود را انجام می‌دهد.



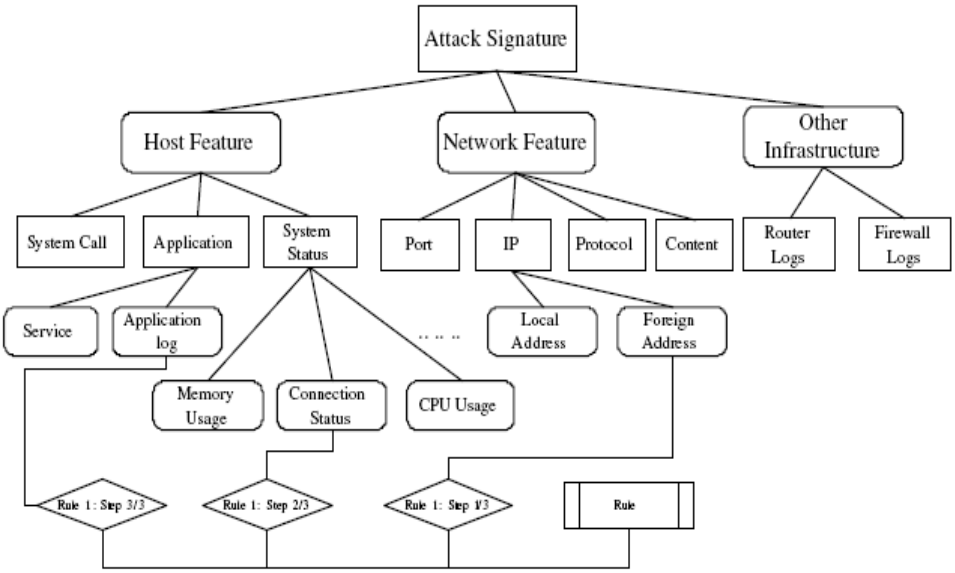
شکل ۴- ۴: آنتولوژی اثر حمله

در این پروژه مدل حمله بر اساس تعاریف معنایی از اجزایی چون *actor*ها (کاربران مجاز و نفوذگران احتمالی)، فعالیت‌ها (قصد و نیت افراد در رابطه با دستورات‌شان) و رفتارها (مفهومی که فعالیت‌های *actor*ها

[۱] *Principal-Subordinate Consequence Tagging Case Grammar*

را دسته‌بندی می‌کند) تشکیل شده است. به همین دلیل در این پروژه آنتولوژی بدست آمده‌ی امضای حمله-ها[۱](*ASO*) هر دو کار کلاس‌بندی نیات افراد و کلاس پیش‌گویی را انجام می‌دهد. طبقه‌بندی اصلی این آنتولوژی در شکل ۴- ۴ نشان داده شده است.

پروژه‌های دیگری که در این زمینه مورد بررسی قرار گرفته یک سیستم تشخیص نفوذ همکاری کننده‌ی مبتنی بر آنتولوژی است*[Yan04]*. آنتولوژی مورد استفاده در طراحی این سیستم بر مبنای بررسی و آنالیز برخی قواعد سیستم‌های تشخیص نفوذ و آسیب‌پذیری‌های امنیتی منتشر شده توسط[۲]*CVE* می‌باشد. این آنتولوژی شامل دو نوع گره می‌باشد: گره‌های*Attribute* و گره‌های *Value*. بخشی از آنتولوژی شرح داده شده در شکل ۴- ۵ نمایش داده شده است. یکی از مزایای استفاده از آنتولوژی در این سیستم این است که به راحتی می‌توان مکان و سنسور مورد نظر برای دستیابی به اطلاعاتی خاص را پیدا کرد. این مسئله در سیستم-هایی که به صورت همکاری کار می‌کنند، بسیار مهم و ضروری است.



شکل ۴- ۵: قسمتی از آنتولوژی مورد استفاده در *[Yan04]*

برای اینکه بتوان با استفاده از آنتولوژی نشان داده شده در شکل ۴- ۵ عمل تشخیص نفوذ را بهتر انجام داد، از یک روش تطبیقی استفاده شده است. در این روش یال‌هایی که بین گره‌های *Value* و پدران آن‌ها قرار

[۱] *Attack Signature Ontology*

[۲] *Common Vulnerabilities and Exposures*

دارند دارای یکسری وزن می‌باشند که نشان دهنده‌ی میزان شباهت گره‌های *Value* که دارای یک پدر هستند، می‌باشد. این وزن‌ها دارای مقادیری بین صفر تا یک است، که مقدار یک نشان دهنده‌ی بیشترین شباهت و مقدار صفر نشان دهنده‌ی عدم شباهت دو گره می‌باشد. در این روش هر یک از این وزن‌ها توسط فرد خبره تعیین شده است، که مقدار هر کدام از آن‌ها با توجه به نتایج بدست آمده و بازخورد سیستم می‌تواند تغییر کند. به این ترتیب در این روش برای هر گره یک بردار در نظر گرفته شده است که نشان دهنده‌ی وزن‌های آن می‌باشد. مثلاً برای گره‌ی N بردارV_N را داریم $V_N(w_1, w_2, \dots, w_n)$که در این بردار w_i نشان دهنده‌ی وزن بین گره‌ی N و برادرش گره‌ی i می‌باشد.

در این روش برای تشخیص اینکه، آیا حالت جدید سیستم نشان دهنده‌ی نفوذ به سیستم است یا نه، می‌بایست میزان شباهت وضعیت فعلی سیستم با وضعیتی که نفوذ در سیستم رخ داده، سنجیده شود. برای انجام این کار هزینه‌ی مسیرهایی که بین گره‌ی *Rule* و گره‌ی *intrusion* که دارای یک پدر می‌باشند، محاسبه شده و در نهایت هزینه‌ی کلی به روش زیر محاسبه می‌شود:

$$\text{path_simi_score}_{\text{path}(i,j)} = V_i(w_1, w_2, ..w_n) * (0_1, 0_2, ..e_j.., 0_n)' \quad (e_j = 1)$$

$$\text{total_score}(N_R, N_I) = \frac{\sum_{path(i,j) \in path_set(N_R, N_I)} \text{path_simi_score}_{\text{path}(i,j)}}{\text{Count}(\text{pathset}(N_R, N_I))}$$

در صورتی‌که هزینه‌ی کلی از یک حد خاص بالاتر باشد می‌توان نتیجه‌گیری کرد که وضعیت فعلی سیستم نیز مربوط به یک نفوذ است و می‌بایست سیستم تشخیص نفوذ هشدار دهد. یکی از عیوب این روش این است که این وزن‌ها تنها توسط یک فرد خبره تعیین شده است و روشی برای بهبود و بهینه کردن وزن‌ها در این روش در نظر گرفته نشده است. ما در این پایان‌نامه از این تکنیک برای بدست آوردن شباهت معنایی استفاده کرده و برخلاف روش شرح داده شده که وزن‌ها توسط فرد خبره تنظیم می‌شوند، سعی نموده‌ایم که وزن‌ها را با استفاده از تکنیک‌های داده‌کاوی بدست آوریم.

آخرین پروژه‌ای که در این زمینه مورد بررسی قرار گرفت در سال ۲۰۰۷ در دانشگاه فردوسی مشهد انجام شده است*[Abd07][Abd09]*. در این پروژه خانم عبدلی از آنتولوژی حملات برای تشخیص نفوذهای کلاس *DoS* در یک سیستم تشخیص نفوذ توزیع شده استفاده نموده است. در این تحقیق با بهره‌گیری از دانش یک

فرد خبره در حوزه‌ی تشخیص نفوذ، ویژگی حملات کلاس *DoS* مشخص می‌شود. سپس این ویژگی‌ها را در قالب مجموعه‌ای از قیود بر روی کلاس‌های آنتولوژی حملات اعمال می‌نماییم. استفاده از روش دستی برای تولید قوانین بسیار وقت‌گیر بوده و به‌خاطر حجم زیاد داده‌های آنالیز مسلماً امکان بروز خطا نیز وجود دارد. علاوه‌بر این در روش دستی بر کوتاه و بهینه بودن قوانین تولیدی توجهی نشده است لذا حجم قیود اعمال شده در آنتولوژی افزایش یافته و این موضوع در کارایی سیستم تشخیص نفوذ نیز تاثیر منفی خواهد داشت. آنتولوژی تولید شده نیز در یک سیستم تشخیص نفوذ چند عامله برای طبقه‌بندی حملات به دو کلاس *DoS* و *NotDoD* مورد استفاده قرار می‌گیرد. در طراحی این سیستم چند عامله نیز به مقیاس پذیری سیستم توجه چندانی نشده و عامل‌های تشخیص هنگام بدست آوردن یک رکورد از وضعیت شبکه آن‌را به عامل مرکزی می‌فرستند. عامل مرکزی نیز با استفاده از آنتولوژی حملات، کلاس این رکورد را تعیین و وضعیت آن‌را به میزبان مربوطه اعلام می‌نماید. برای ارزیابی این سیستم از مجموعه داده‌ای *KDD cup* استفاده شده است. سیستم پیشنهادی در *[Abd09]* نرخ تشخیص بالایی داشته ولی به‌خاطر کامل نبودن آنتولوژی حملات، نرخ هشدار غلط آن نسبتاً زیاد می‌باشد که این امر از نقایص سیستم محسوب می‌شود. به‌خاطر گستردگی موضوع و پیچیدگی ساخت آنتولوژی، این تحقیق با تمام توانایی‌هایش ناتمام مانده بود. در این پایان‌نامه قصد داریم با کمک گرفتن از تکنیک‌های داده‌کاوی و تشابه معنایی، علاوه بر تولید یک آنتولوژی کامل برای حملات و رفتار نرمال، از این آنتولوژی در یک سیستم همکارانه‌ی تشخیص نفوذ استفاده نمایم.

۴-۴- خلاصه

بحث وب‌معنایی و استفاده از معنا و مفهوم در وب و قابل استفاده نمودن اطلاعات موجود در آن برای ماشین-ها و سیستم‌ها بحث نسبتاً جدید و تازه‌ای در دنیای علوم کامپیوتری و مهندسی کامپیوتر می‌باشد. این فصل به معرفی اجمالی وب‌معنایی، ابزارها و تکنیک‌های مورد استفاده آن پرداخته است، در ادامه نیز بحث استفاده از این مفاهیم و تکنیک‌ها را در مباحث مربوط به سیستم‌های تشخیص نفوذ مطرح کرده و در این زمینه پروژه‌ها و تحقیقاتی را که در سال‌های اخیر انجام شده مورد بررسی قرار داده است.

فصل پنجم

« طراحی آنتولوژی حملات کامپیوتری»

۵–۱– مقدمه

این فصل ابتدا نحوه‌ی طراحی آنتولوژی‌ها و نحوه‌ی عملی ایجاد آن‌ها به‌اختصار شرح داده شده است و در ادامه‌ی فصل نحوه‌ی طراحی و استخراج آنتولوژی پیشنهادی در این پایان نامه یا همان "آنتولوژی حملات کامپیوتری" مورد بحث و بررسی قرار گرفته است همچنین جزئیات نحوه‌ی استخراج اطلاعات مورد نیاز برای طراحی آنتولوژی "حملات کامپیوتری" و فازهای مختلف از ایجاد آنتولوژی به تفصیل شرح داده شده است.

۵–۲– محیط طراحی آنتولوژی

برای طراحی آنتولوژی مورد نظر از نرم‌افزار *Protégé* استفاده شده است. این نرم‌افزار یک نرم‌افزار منبع باز رایگان مبتنی بر پایگاه دانش می‌باشد*[Pro09]*، همچنین برای بررسی میزان سازگاری اجزای آنتولوژی با یکدیگر از نرم‌افزار *Racer* به عنوان نرم‌افزار "استدلال کننده" استفاده شده است*[Rac09]*. طراحی آنتولوژی با استفاده از نرم‌افزار *Protégé* در دو بستر مختلف قابل انجام است:

- محیط ویرایشی *Protégé-Frames*
- محیط ویرایشی *Protégé-OWL*

ویرایشگر *Protégé-Frames* به طراحان این امکان را می‌دهد، تا آنتولوژی‌هایی مبتنی بر قاب[۲]که مطابق با پروتکل[۳]*OKBC* می‌باشد، را طراحی کنند. در این مدل، هر آنتولوژی تشکیل شده از مجموعه‌ای از کلاس‌ها که به واسطه‌ی مجموعه‌ای از *slot*ها، ویژگی‌ها و روابط ما بین این کلاس‌ها مشخص شده است، و برای هر کلاس تعدادی نمونه در نظر گرفته شده که مقادیری خاص را برای هر یک از ویژگی‌های آن کلاس مشخص می‌کنند.

اما ویرایشگر *Protégé-OWL* به طراحان این امکان را می‌دهد، تا آنتولوژی‌هایی مبتنی بر وب‌معنایی طراحی کنند. البته لازم به ذکر است که محیط *Protégé* این امکان را به طراحان می‌دهد تا برای طراحی آنتولوژی مورد نظر خود از محیط ویرایشی مبتنی بر قاب استفاده کرده و پس از اتمام مرحله‌ی طراحی، آنتولوژی خود

---

[۱] *Reasoner*

[۲] *Frame-Based*

[۳] *Open Knowledge Base Connectivity Protocol*

را به مدل *OWL* تبدیل کنند. در این صورت آنتولوژی طراحی شده نمی‌تواند از تمام قابلیت‌های محیط *OWL* استفاده کند، زیرا طراحی اصلی آنتولوژی در محیط مبتنی بر قاب صورت گرفته است.

آنتولوژی طراحی شده با استفاده از نرم‌افزار *Protégé* قابل تبدیل به فرمت‌های مختلفی چون: *RDF(S)*، *OWL*، *XML Schema* و... می‌باشد. نرم‌افزار *Protégé* مبتنی بر *Java* و قابل توسعه می‌باشد، این نرم‌افزار به دلیل فراهم کردن محیط *plug-and-play* انعطاف‌پذیری بالایی دارد.

در این پایان‌نامه به منظور حداکثر استفاده از امکانات *OWL* از محیط ویرایشی *Protégé-OWL* استفاده شده است. پس از طراحی آنتولوژی "حملات کامپیوتری" در محیط *Protégé* از فایل *OWL* تولید شده، برای دستیابی به سایر اهداف مورد نظر در این پایان‌نامه استفاده شده است. در فصل ششم نحوه‌ی استفاده از آنتولوژی طراحی شده و همچنین محیط طراحی شده برای تست آنتولوژی مورد نظر به تفصیل مورد بررسی قرار خواهد گرفت.

۵– ۳– طراحی و ایجاد آنتولوژی

به طور کلی دو روش اصلی برای طراحی آنتولوژی وجود دارد*[Nat01]*:

- استفاده از آنتولوژی‌های طراحی شده توسط سایرین در دامنه‌های مورد نظر و سپس تغییر و تکمیل آن‌ها در جهتی که مناسب برای کاربرد مورد نظر طراح باشد.
- استفاده از طبقه‌بندی‌های موجود در دامنه‌های مورد نظر و طراحی آنتولوژی مورد نظر بر مبنای این طبقه‌بندی‌ها.

با توجه به اینکه که در حال حاضر آنتولوژی کامل و جامعی در زمینه‌ی حملات کامپیوتری وجود ندارد، در این پایان‌نامه از روش دوم برای طراحی و ایجاد آنتولوژی استفاده شده و کلیه‌ی مراحل ایجاد آنتولوژی از ابتدای فاز طراحی به‌طور کامل انجام شده است. مراحل عملی طراحی و ایجاد یک آنتولوژی عبارت است از*[Nat01]*:

۱. تعریف کلاس‌ها در آنتولوژی
۲. مرتب کردن سلسله مراتب کلاس‌ها در آنتولوژی

۳. تعریف ویژگی‌ها و خصایص و همچنین معین کردن مقادیر مجاز برای آن‌ها

۴. مقداردهی به خصایص برای نمونه‌های تعریف شده در آنتولوژی

انجام هر یک از این مراحل و جمع‌آوری اطلاعات برای آن‌ها، استخراج دانش و استنتاج روابط ما بین‌شان جزء امور بسیار پیچیده در طراحی آنتولوژی می‌باشد، البته تست و ارزیابی آنتولوژی طراحی شده نیز دارای پیچیدگی‌های خاص خود است.

توجه به این نکته ضروری است که آنتولوژی مدلی از واقعیات و حقایق است و مفاهیم در آنتولوژی می‌بایست نشان دهنده‌ی این حقایق باشند. برای دستیابی به این هدف بهتر است ابتدا نسخه‌ی اولیه‌ای از آنتولوژی مورد نظر تهیه شود و سپس با به‌کاربردن این آنتولوژی در برنامه‌های کاربردی مناسب و همچنین کمک گرفتن از افراد متخصص در حوزه‌ای که آنتولوژی مورد نظر در آن تعریف شده است، آنتولوژی ایجاد شده را ارزیابی و اشکال‌زدایی کرده و اندک اندک آن را بهبود بخشیده و کامل کرد، در واقع چرخه‌ی تکامل برای آنتولوژی در نظر گرفت.

۵- ۳- ۱- تعریف کلاس‌ها در آنتولوژی

اولین مرحله از فاز ایجاد آنتولوژی، مشخص کردن دامنه و قلمروی آن آنتولوژی می‌باشد. سئوالاتی از قبیل مشخص شدن حوزه‌ای که آنتولوژی قرار است آن را پوشش دهد، دلیل ساختن آنتولوژی و یا اینکه آنتولوژی مورد نظر برای چه مدل از سوالات می‌بایست پاسخ داشته باشد و... به مشخص کردن دامنه و قلمرو آنتولوژی مورد طراحی کمک می‌کنند. در این پایان‌نامه هدف این است که از آنتولوژی طراحی شده در فاز تشخیص نفوذ، در سیستم‌های تشخیص نفوذ استفاده شود. بنابراین می‌بایست آنتولوژی طراحی شود که پوشش دهنده حملات معمول واقع شده در شبکه‌ها و سیستم‌های کامپیوتری بوده و در رابطه با آن‌ها اطلاعات داشته باشد. این آنتولوژی باید بتواند ویژگی‌ها و شرایط وقوع حمله در شبکه‌ها را پوشش داده و به تشخیص حملات کمک کند. برای اینکه سیستم تشخیص نفوذ قدرت بیشتری داشته و توانایی تشخیص نفوذهای جدید را نیز داشته باشد علاوه بر رفتارهای نفوذی بایستی الگوی رفتارهای نرمال نیز در دل این آنتولوژی گنجانده شوند. با بهره-گیری از این تکنیک سیستم به یک سیستم ترکیبی تبدیل شده که از مزایای هر دو سیستم تشخیص نفوذ

سوء استفاده و رفتارهای غیرعادی بهره‌مند می‌باشد. به عبارت دیگر نفوذهایی را که دارای الگوی شناخته شده‌ای باشند با دقت و سرعت زیاد تشخیص داده و همچنین قادر به تشخیص نفوذهای جدید نیز می‌باشد.

۵- ۳- ۲- مرتب کردن سلسله مراتب کلاس‌ها در آنتولوژی

در فاز دوم و قبل از شروع به طراحی آنتولوژی بهتر است آنتولوژی‌های آماده و موجود در دامنه‌ی مورد نظر طراح مورد بررسی قرار گرفته و در صورت مناسب بودن، از آن‌ها به عنوان آنتولوژی اولیه استفاده کرده و با توجه به اهداف مورد نظر آن‌ها را تکمیل نمود. البته بررسی و درک طراحی که فرد دیگری انجام داده کمی سخت است، اما وقتی که قرار است آنتولوژی طراحی‌شده با سایر کاربردهایی که وابسته به این آنتولوژی هستند در تعامل باشد، استفاده از آنتولوژی‌های طراحی شده‌ی موجود امری ضروری به نظر می‌رسد. کتابخانه‌هایی از آنتولوژی‌های قابل استفاده در سطح وب وجود دارد[۱] اما همان‌طور که پیشتر نیز اشاره شد در حوزهٔ حملات کامپیوتری کار خاصی در زمینه‌ی طراحی آنتولوژی انجام نشده است. بنابراین آنتولوژی اولیه‌ای نیز در این زمینه وجود ندارد.

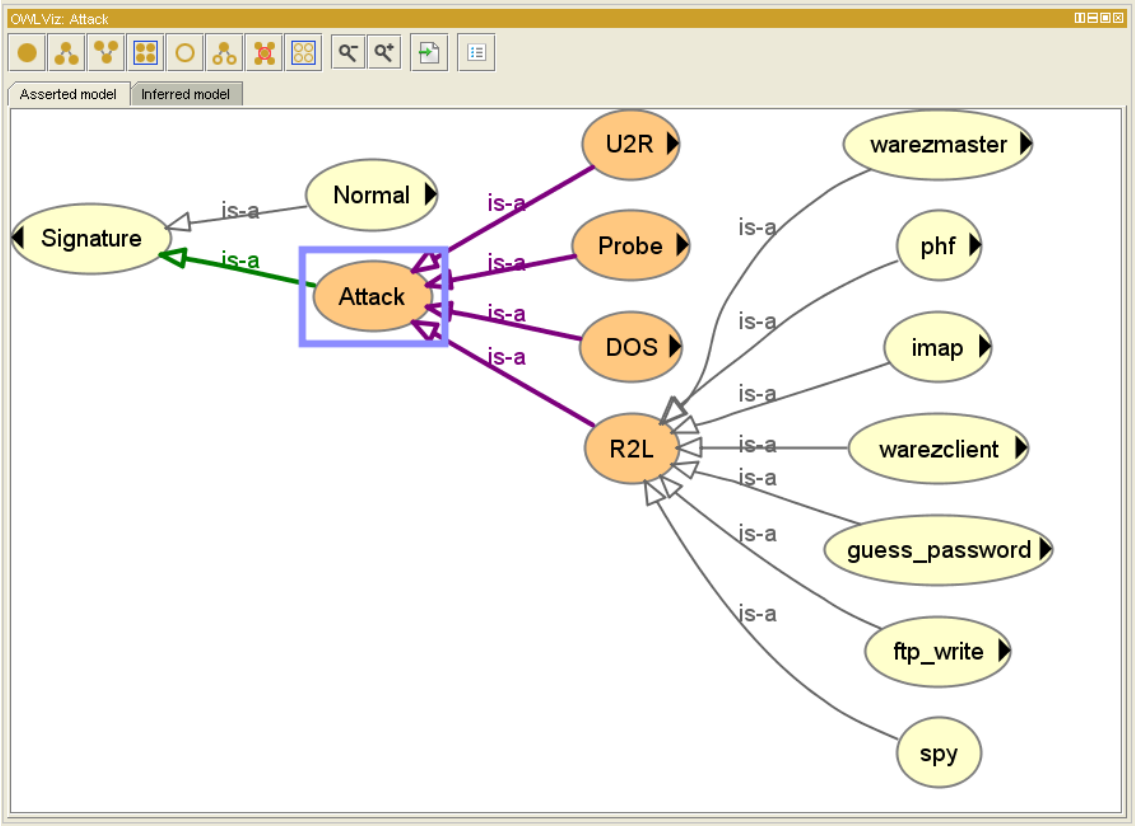
برای تعیین سلسله مراتب کلاس‌های آنتولوژی روش‌های مختلفی وجود دارد: روش بالا به پایین، روش پایین به بالا و روشی که ترکیبی از این دو روش می‌باشد*[Nat01]*. در روش بالا به پایین ابتدا عمومی‌ترین مفاهیم مورد بررسی قرار می‌گیرند. در روش پایین به بالا پروسه‌ی طراحی با تعریف کلاس‌های خیلی خاص دامنه آغاز شده و کم‌کم به سمت کلاس‌های اصلی‌تر و عمومی‌تر پیش می‌رود. در روش آخر پروسه‌ی طراحی و ساخت، ترکیبی از دو روش بالا به پایین و پایین به بالا خواهد بود.

در این پایان‌نامه نیز در طراحی آنتولوژی "حملات کامپیوتری" از روش بالا به پایین استفاده شده است. در "آنتولوژی حملات کامپیوتری" مفهوم *Signature* کلی‌ترین مفهوم در دامنه‌ی تشخیص حملات کامپیوتری می‌باشد، که می‌تواند دارای زیرشاخه‌های *Attack* و *Normal* باشد. زیر شاخه‌ی *Attack* در واقع شامل امضای تمام نفوذهای شناخته شده و موجود در داده‌های آموزشی *NSL-KDD* بوده که خود دارای چهار زیرشاخه‌ی کلی *DoS*، *U2R*، *R2L* و *Probe* می‌باشد. هر کدام از این کلاس‌های حملات نیز خود دارای انواع مختلفی

---

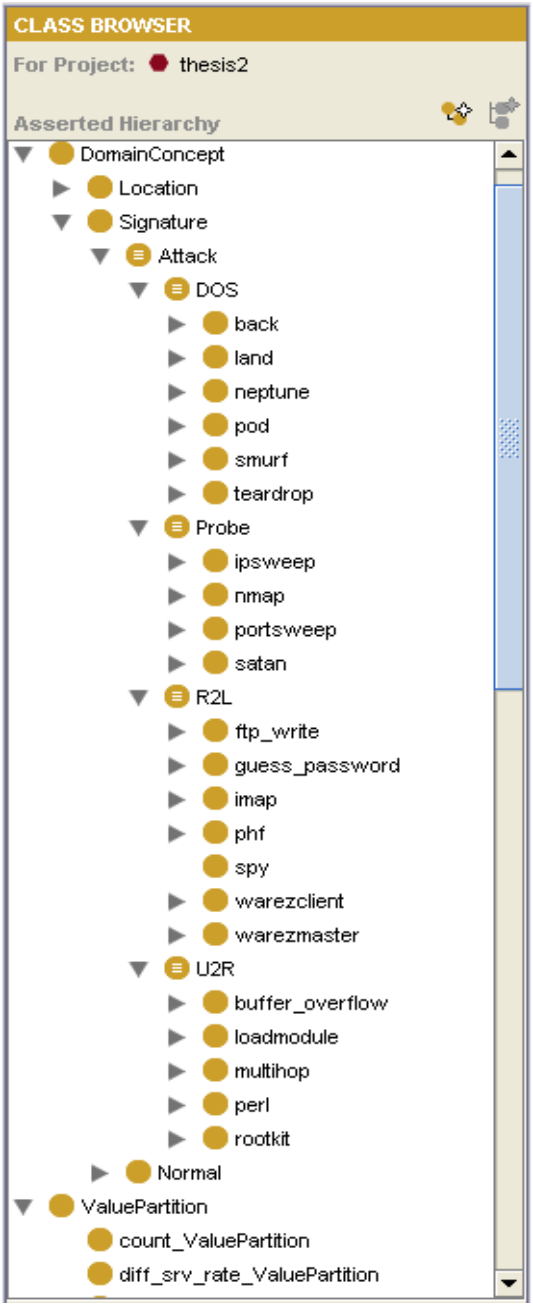
[۱] *I. http://www.ksl.stanford.edu/software/ontolingua/*
II. http://www.daml.org/ontologies/

می‌باشند، مثلاً کلاس حملات *R2L* دارای زیر کلاس‌های *ftp_write*، *guess_password*، *imap*، *phf*، *spy*، *warezclient* و *warezmaster* می‌باشد. در این مرحله دسته‌بندی اصلی حملات کامپیوتری انجام شده است، در ادامه نیز به بررسی و تکامل حملات و زیرشاخه‌های حملات پرداخته می‌شود. در شکل ۵- ۱ شمایی از کلاس حملات *R2L* و زیرکلاس‌های آن نشان داده شده است. شکل ۵- ۲نیز نشان دهنده‌ی اسکلت اصلی آنتولوژی "حملات کامپیوتری" می‌باشد. لیست کامل کلاس‌های حملات مختلف موجود در مجموعه داده‌های *NSL-KDD* در جدول ۲ از ضمیمه‌ی الف بطور کامل ذکر شده است.

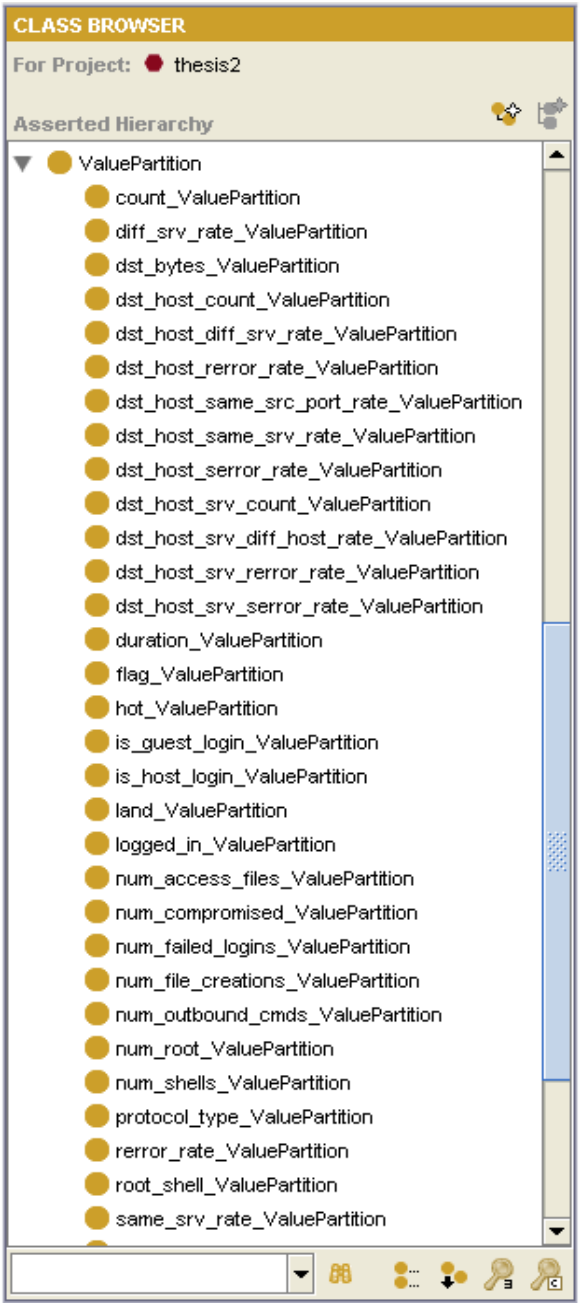


شکل ۵- ۱: شمایی از کلاس حملات *R2L* و زیرکلاس‌های آن

در تعریف کلاس‌های آنتولوژی کلاس عمومی دیگری تحت عنوان *ValuePartition* برای بیان ویژگی‌ها و ارتباطات حملات مختلف با یکدیگر تعریف شده است. هدف از طراحی این کلاس کمک گرفتن برای بیان ویژگی‌ها و ارتباطات سایر کلاس‌ها با یکدیگر می‌باشد. کلاس‌هایی مثل کلاس *ValuePartition* در واقع جزء آنتولوژی اصلی نبوده و تنها برای کمک به طراحی آنتولوژی اصلی و مقداردهی ویژگی‌های کلاس‌های مختلف موجود در آنتولوژی تعریف می‌شوند.



شکل ۵- ۲: اسکلت اصلی آنتولوژی "حملات کامپیوتری"



شکل ۵- ۳ : ساختار کلاس‌های *ValuePartition*

در آنتولوژی طراحی شده در این پایان‌نامه کلاس *ValuePartition* دارای ۴۱ زیر کلاس بوده که هر کدام از آن‌ها دارای نمونه‌هایی هستند که این نمونه‌ها مقادیر مجاز برای ویژگی‌های ۴۱ گانه‌ی اتصالات مربوط به حملات کامپیوتری می‌باشند(شکل ۵- ۳).

از کلاس‌های *ValuePartition* موجود در آنتولوژی پیشنهادی می‌توان به *service_ValuePartition*، *flag_ValuePartition*، *protocol_type_ValuePartition* و... اشاره کرد که در بخش بعد دلیل ایجاد این کلاس‌ها و چگونگی برقراری ارتباط مابین سایر کلاس‌ها توسط یک مثال به تفصیل شرح داده می‌شود.

۵- ۳- ۳- تعریف ویژگی‌ها و خصایص و معین کردن مقادیر مجاز برای آن‌ها

در مرحله‌ی سوم طراحی می‌بایست اصطلاحات و موارد مهم و حساس در آنتولوژی لیست شوند. مثلاً در رابطه با حملات کامپیوتری ویژگی‌هایی چون "پروتکل مورد استفاده"، "سرویس مورد استفاده" و... می‌توانند جزء موارد اصلی و مهم در رابطه با حملات در آنتولوژی "حملات کامپیوتری" باشند. در این مرحله مقادیر مجاز برای هرکدام از این ویژگی‌ها نیز بایستی مورد بررسی قرار گرفته و ویژگی‌ها و خصایص کلاس‌های آنتولوژی تعیین شوند. تنها تعریف کلاس‌ها برای داشتن آنتولوژی کافی نیست، برای اینکه آنتولوژی بتواند اهداف مورد نظر طراحی را برآورده کند و پاسخ مناسبی برای سوال‌ها و پرس و جوها داشته باشد می‌بایست خصایص و ویژگی‌های کلاس‌های آن تعیین شده و محدودیت‌های آن‌ها نیز مشخص شود.

به عنوان مثال در آنتولوژی "حملات کامپیوتری" از ۲۶۴۶ رکورد اتصال مربوط به کلاس حملات *Smurf* تمامی آن‌ها دارای سرویس *ecr_i* بوده و تعداد بایت‌های داده‌ای فرستاده شده از مبدا به مقصد[۱] بیشتر از ۵۲۰ بایت می‌باشد و یا اینکه در کلاس حملات *Land* حتماً می‌بایست خصوصیت[۲] *land* برابر با یک باشد.

1) *(service = ecr_i) and (src_bytes >= 520) => class=smurf*
2) *(land = 1) => class=land*

اگر داده‌های آزمایشی مورد استفاده در این تحقیق بسیار کمتر از مقدار کنونی بود یک شخص خبره با آنالیز آن‌ها و انجام محاسبات آماری بر روی آن‌ها می‌توانست مجموعه‌ای از این قوانین را بدست آورد. همان‌طور که در تحقیقی که توسط خانم عبدلی انجام پذیرفت، بیرون‌کشیدن این قوانین از دل داده‌های آزمایشی توسط یک فرد خبره انجام شده‌بود*[Abd09]*. این روش علاوه بر وقت‌گیر بودن، دقت پایینی نیز خواهد داشت و در مجموعه‌ی داده‌های آزمایشی عظیم روش دستی عملاً غیر ممکن می‌باشد. مجموعه‌ی این عوامل باعث گرایش

---

[۱] *src_bytes*

[۲] *connection from/to the same host/port (yes/no)*

ما به سمت تکنیک‌های داده‌کاوی گردید. همانطور که در فصل سوم نیز بیان شد تکنیک‌های داده‌کاوی زیادی وجود دارد که امکان استخراج قوانین را از داده‌های برچسب زده فراهم می‌کنند. ما در این تحقیق روش *RIPPER* را برای این‌منظور انتخاب نموده‌ایم.

از طرف دیگر در این مطالعه دو مجموعه داده‌های آموزشی و آزمایشی *NSL-KDD* که برای هر رکورد اتصال دارای۴۱ خصیصه و یک برچسب که مشخص کننده‌ی نوع آن رکورد می‌باشد، مبنایی برای یادگیری قوانین، مقایسه و انجام آزمایشات بوده است. در این مرحله از ایجاد آنتولوژی "حملات کامپیوتری" با استفاده از اعمال روش‌های مختلف داده‌کاوی بر روی داده‌های آموزشی *NSL-KDD*، ویژگی‌ها و خصایص هر یک از کلاس‌ها و مقادیر مجاز آن‌ها تعیین شده است. در ادامه پس از بررسی مجموعه‌ی داده‌های *NSL-KDD*، فرایند استخراج قوانین *RIPPER* از مجموعه‌ی داده‌های آزمایشی *NSL-KDD* مورد بررسی قرار خواهد گرفت.

۵- ۳- ۳- ۱- مجموعه داده‌های *KDD'99*

مجموعه داده‌های *KDD cup 99* شامل ۴۱ خصیصه استخراج شده برای هر اتصال می‌باشد که یک برچسب وضعیت اتصال را که نرمال یا یک حمله از کلاسی خاص است، مشخص می‌کند. این خصیصه‌ها به طور کلی دارای چهار فرم پیوسته[۱] گسسته[۲]و سمبولیک[۳]با بازه وسیعی از مقادیر هستند که به طور کلی به چهار دسته تقسیم می‌شوند *[KDD99]*:

- دسته اول شامل خصیصه‌های ذاتی[۴]یک اتصال هستند، که به نوبه خود شامل خصیصه‌های پایه اتصالهای *TCP* است. مدت یک اتصال، نوع پروتکل (*TCP, UDP, ICMP*) و نوع سرویس مورد استفاده (*telnet , http, ...*) نمونه‌هایی از این خصیصه‌ها هستند.
- خصیصه‌های محتوایی[۱] بخشی از خصیصه‌های یک اتصال هستند که از طریق وارسی بخش داده‌ای بسته‌های *TCP* بدست می‌آیند. به عنوان مثال تعداد *login*های شکست خورده[۲]نمونه‌ای از این خصیصه‌ها است.

---

[۱] *Continuous*
[۲] *Discrete*
[۳] *Symbolic*
[۴] *Intrinsic*

- خصیصه‌های میزبان یکسان ، اتصالاتی را که در دو ثانیه گذشته دارای مقصد یکسان با اتصال جاری بوده‌اند بررسی می‌کند و مقادیر آماری که به رفتار پروتکل، سرویس و غیره وابسته است را بدست می‌آورند.
- خصیصه‌های سرویس یکسان مشابه[۳] خصیصه‌هایی هستند که اتصالاتی که در دو ثانیه گذشته سرویس یکسانی با اتصال جاری دارند را بررسی می‌کند. و مقادیر آماری خاصی را برای آن‌ها محاسبه می‌کنند

به همین ترتیب حملات در این مجموعه به چهار دسته اصلی تقسیم می‌شوند:

- حملات جلوگیری از سرویس *DoS*[۴] این دسته از حملات سعی می‌کنند منابع محاسباتی یا حافظه را آنقدر مشغول کنند تا کاربران نتوانند از حقوق مشروع خود برای استفاده از این منابع بهره گیرند.
- حملات کاربر به هسته سیستم *U2R*[۵] در این دسته از حملات، مهاجم با استفاده از حفره‌های امنیتی سیستم به حقوق کاربر ارشد سیستم دسترسی پیدا کرده و فعالیت‌های غیر قانونی‌اش را انجام می‌دهد.
- حملات سیستم دور به ماشین محلی *R2L*[۶] دسترسی غیر مجاز از طریق حفره‌های امنیتی به یک ماشین با استفاده از یک ماشین راه دور، در این دسته قرار می‌گیرد.
- حملات پویشی[۷] در این دسته از حملات مهاجم به بررسی میزبان‌ها و پرت‌های مختلف می‌پردازد تا بتواند حفره‌های امنیتی آن‌ها را کشف کند و دراصل این دسته از حملات زمینه‌سازی برای حملات دیگر هستند.

---

[۱] *Content Feature*
[۲] *Failed Login Attemp*
[۳] *Similar Same Service*
[۴] *Denial of Service*
[۵] *User to Root*
[۶] *Remote to Local*
[۷] *Probe*

۵- ۳- ۳- ۲- مجموعه داده‌های آزمایشی *NSL-KDD*

برای بهبود برخی مشکلات ذاتی موجود در مجموعه داده‌های آزمایشی *KDD* مجموعه داده‌های آزمایشی *NSL-KDD* پیشنهاد شده است[*Tav09*]. اگر چه این مجموعه جدید از برخی مشکلات بیان شده مبری نبوده و نمی‌تواند یک نمونه واقعاً کامل و عملی برای رکوردهای اتصال[۱]شبکه‌های موجود باشد، با این وجود این مجموعه می‌تواند توسط محققان به عنوان یک مجموعه مناسب برای مقایسه سیستم‌های کشف نفوذ مختلف مورد استفاده قرار گیرد.

a) اصلاحات مجموعه *NSL-KDD*

مجموعه داده‌های *NSL-KDD* نسبت به مجموعه داده‌های *KDD'99* دارای مزایای زیر می‌باشد:

- در مجموعه آموزش داده‌های تکراری وجود ندارد و طبقه بندها براساس داده‌های تکرار شده بایاس نمی‌شوند.
- در برخی روش‌ها یادگیرنده‌ها برای داده‌های تکراری نرخ تشخیص بالاتری دارند. در مجموعه آزمایشی *NSL-KDD* هیچ داده تکراری وجود نداشته و کارآیی یادگیرنده‌ها براساس این داده‌های تکراری بایاس نمی‌شود.
- علاوه بر این تعداد رکوردهای آموزش[۲]و تست[۳]در مجموعه *NSL-KDD* منطقی است. این مزیت باعث می‌شود که برای استفاده از این مجموعه در آزمایشات مختلف، احتیاج به انتخاب یک زیر مجموعه کوچکتر نباشد. متعاقبا نتایج ارزیابی محققان مختلف سازگار و قابل مقایسه خواهد بود.

b) مشاهدات آماری مجموعه داده‌های *KDD'99*

یکی از بزرگترین عیوب مجموعه داده‌های *KDD* تعداد بسیار زیاد رکوردهای زائد و تکراری است. این امر باعث می‌شود الگوریتم‌های یادگیری براساس داده‌های زائد بایاس شده و مانع از یادگیری داده‌های غیر زائدی که برای شبکه‌ها خطرناک می‌باشند، شوند.

---

[۱] *Connection*

[۲] *Ttrain*

[۳] *Test*

به عنوان مثال داده‌های مربوط به حملات *U2R* و *R2L* در این مجموعه بسیار کم بوده و در امر آموزش الگوریتم‌های مختلف تاثیر چندانی ندارند. این افزونگی در مرحله ارزیابی الگوریتم‌های مختلف نیز باعث می-شود الگوریتم‌هایی که نتایج بهتری برای داده‌های تکراری دارند، دارای نرخ تشخیص بیشتری شوند که این امر مطلوب نیست(جدول ۵- ۱ و جدول ۵- ۲).

برای داده‌کاوی در داده‌های عظیم نرم افزارهای متعددی طراحی شده که بسیاری از روش‌های داده‌کاوی را پیاده‌سازی کرده و کار با آن‌ها نیز بسیار راحت می‌باشد. از جمله معروفترین و پرکاربردترین آن‌ها می‌توان به *Weka* و *RapidMiner* اشاره نمود. در این تحقیق از *RapidMiner* برای بدست آوردن ویژگی‌های حملات و نفوذهای کامپیوتری مختلف با استفاده از تکنیک‌های داده‌کاوی، استفاده شده است.

جدول ۵- ۱ : آمار رکوردهای زائد در مجموعه آموزش *KDD*

	Original Records	***Distinct Records***	***Reduction Rate***
Attack	*3,925,650*	*262,178*	*93.32%*
Normal	*972,781*	*812,814*	*16.44%*
Total	*4,898,431*	*1,074,992*	*78.05%*

جدول ۵- ۲: آمار رکوردهای زائد در مجموعه تست *KDD*

	Original Records	***Distinct Records***	***Reduction Rate***
Attack	*250,436*	*29,378*	*88.26%*
Normal	*60,591*	*47,911*	*20.92%*
Total	*311,027*	*77,289*	*75.15%*

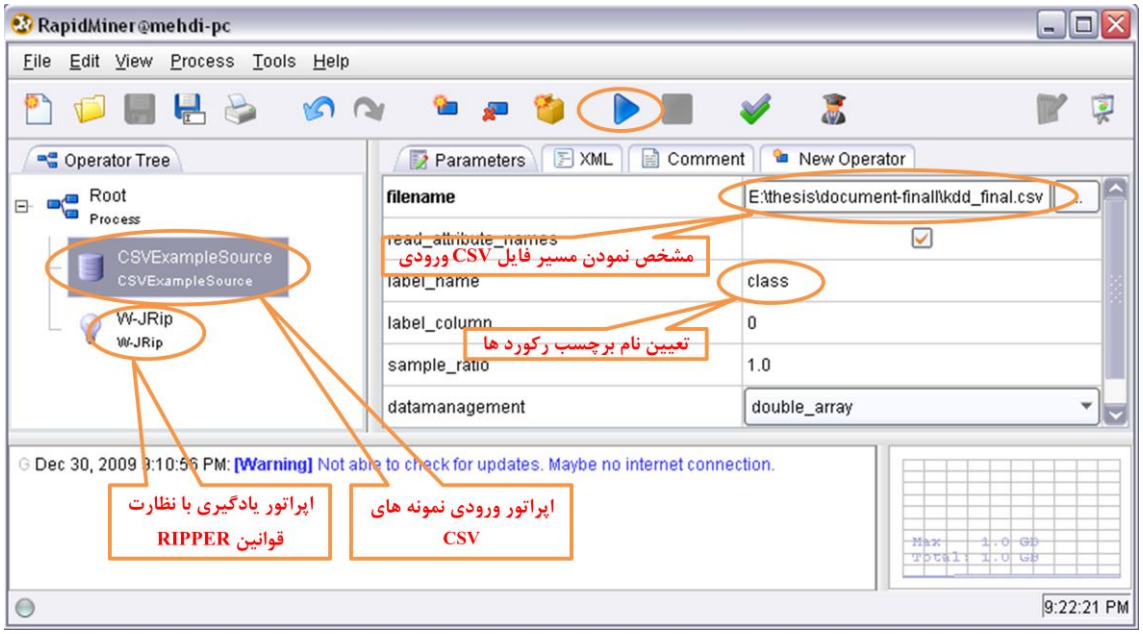
۵- ۳- ۳- ۳- یادگیری از داده‌های *NSL-KDD* با استفاده از قوانین *RIPPER*

در این تحقیق برای استخراج قوانین از مجموعه‌ی داده‌های آموزشی پس از انجام آزمایشات متعدد و بررسی روش‌های مختلف، روش *RIPPER* برای یادگیری قوانین انتخاب شد. در فصل سوم دلایل انتخاب *RIPPER* به عنوان الگوریتم داده‌کاوی یادگیری قانون بیان شد.

برای داده‌کاوی در داده‌های عظیم نرم افزارهای متعددی طراحی شده که بسیاری از روش‌های داده‌کاوی را پیاده‌سازی کرده و کار با آن‌ها نیز بسیار راحت می‌باشد. از جمله معروفترین و پرکاربردترین آن‌ها می‌توان به

Weka و *RapidMiner* اشاره نمود. در این تحقیق از *RapidMiner* برای بدست آوردن ویژگی‌های حملات و نفوذهای کامپیوتری مختلف با استفاده از تکنیک‌های داده‌کاوی، استفاده شده است.

برای شروع لازم است که فایل متنی مربوط به اتصالات مختلف به یکی از فرمت‌های *CSV* و یا *Arff* تبدیل شود زیرا نرم‌افزار *RapidMiner* فقط فایل‌هایی را که در یکی از این فرمت‌ها باشند، می‌شناسد. در اینجا برای ایجاد یک فایل *CSV* یک برنامه‌ی ساده به زبان *C++* نوشته شد که فایل متنی اتصالات مجموعه داده‌های آموزشی برچسب خورده‌ی *NSL-KDD* را خوانده و فایل *CSV* مربوطه را با قالب مورد نظر ایجاد می‌نماید.



شکل ۵- ۴: تنظیمات مربوطه در *RapidMiner*

پس از آن در نرم‌افزار *RapidMiner* در ریشه‌ی درخت عملگرها[۱] یک عملگر ورودی نمونه‌های *CSV* و یک عملگر *RIPPER10* ساخته و پس از تعیین مسیر فایل *CSV* و نام فیلد مربوط به برچسب رکوردها *(class)* نرم‌افزار را اجرا شد تا نتایج حاصل شود (شکل ۵- ۴). نتیجه این داده کاوی‌ها مجموعه‌ای از قوانینی بوده که ویژگی‌های حملات مختلف را در قالب تعدادی قانون ساده و قابل فهم ارائه می‌نماید. در شکل ۵- ۵ چند نمونه از این قوانین نمایش داده می‌شود و نتیجه‌ی کامل این داده‌کاوی در ضمیمه‌ی ب آورده شده است.

[۱] *OperatorTree*

1) *(num_shells >= 1) and (dst_host_same_src_port_rate <= 0.01) => class=perl (4.0/1.0)*
2) *(root_shell >= 1) and (src_bytes <= 51) and (service = http) => class=phf (4.0/0.0)*
3) *(num_file_creations >= 1) and (dst_host_srv_count <= 1) and (num_shells >= 1) => class=multihop (3.0/1.0)*
4) *(dst_host_count <= 2) and (service = login) => class=ftp_write (2.0/0.0)*
5) *(dst_host_count <= 2) and (service = ftp_data) and (dst_host_srv_count >= 84) and (dst_host_srv_count <= 85) => class=ftp_write (2.0/0.0)*
6) *(num_file_creations >= 1) and (src_bytes <= 116) and (src_bytes >= 104) => class=ftp_write (2.0/0.0)*

…

شکل ۵- ۵: چند نمونه از قوانین *RIPPER* مستخرج از داده‌کاوی

قوانین مستخرج از داده‌کاوی در واقع در هر دو فاز سوم (تعریف ویژگی‌ها و خصایص و معین کردن مقادیر مجاز برای آن‌ها) و چهارم (مقداردهی به خصایص برای نمونه‌های تعریف شده) مورد استفاده قرار می‌گیرند. پس از تولید قوانین توسط *RapidMiner* تمام شرط‌های تک‌تک قانون‌ها (قسمت مقدم قانون‌ها) به عنوان مقادیر مجاز برای آن خصیصه در نظر گرفته‌شده و در بخش *ValuePartition* در زیر کلاس مربوطه به‌عنوان یک نمونه اضافه می‌شود. به عنوان مثال قانون زیر را در نظر بگیرید:

(service = ecr_i) and (src_bytes >= 520) => class=smurf

این قانون دارای دو شرط یا مقدم *(service = ecr_i)* و *(src_bytes >= 520)* بوده و حداکثر مقدار ویژگی *src_bytes* برابر با *1379963888* می‌باشد. لذا برای ویژگی *service* در بخش *ValuePartition* مقدار *ecr_i* به‌عنوان یک نمونه‌ی مجاز در کلاس *service_ValuePartition* افزوده شده و برای ویژگی *src_bytes* نیز در بخش *ValuePartition* مقدار زیر به‌عنوان یک نمونه‌ی مجاز در کلاس *src_bytes_ValuePartition* افزوده می‌شود.

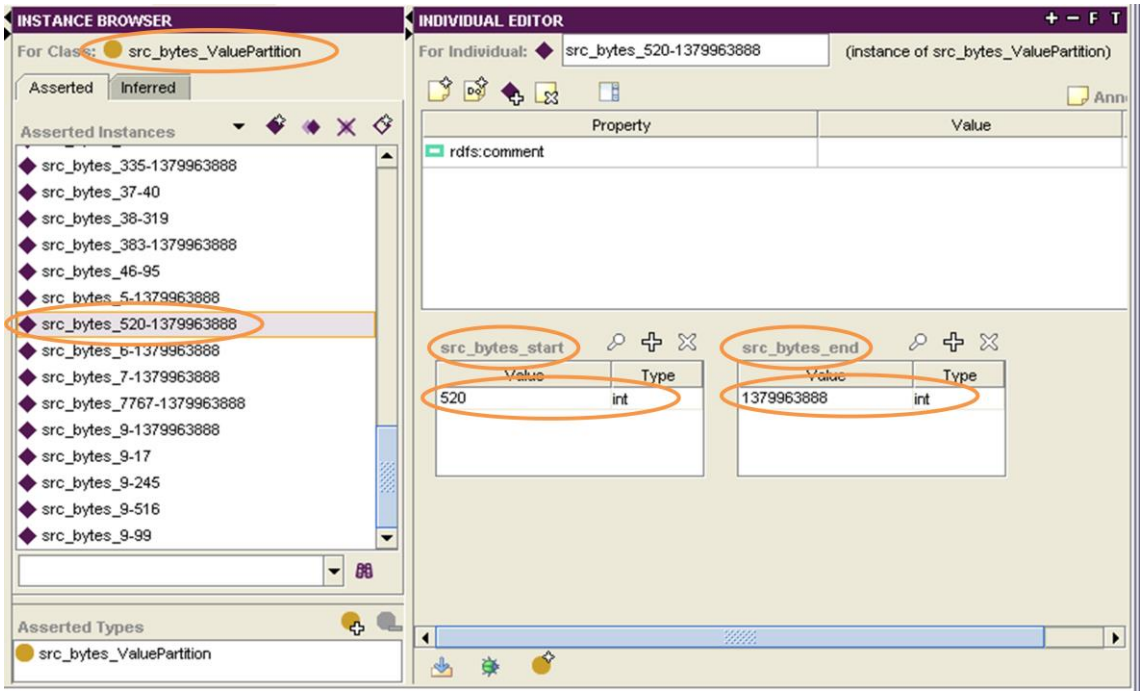
(520 <= src_bytes<= 1379963888)

نرم افزار *Protégé* امکان تعریف بازه را برای کاربران فراهم نمی‌آورد و از آنجا که برای طراحی آنتولوژی استفاده از یک بازه برای ویژگی‌های با مقادیر عددی ضروری می‌باشد لذا برای کلاس‌های *ValuePartition* که دارای مقادیر عددی هستند ونیاز به تعریف بازه دارند دو ویژگی نوع داده‌ای[۱]تعریف می‌شود که مربوط به ابتدا

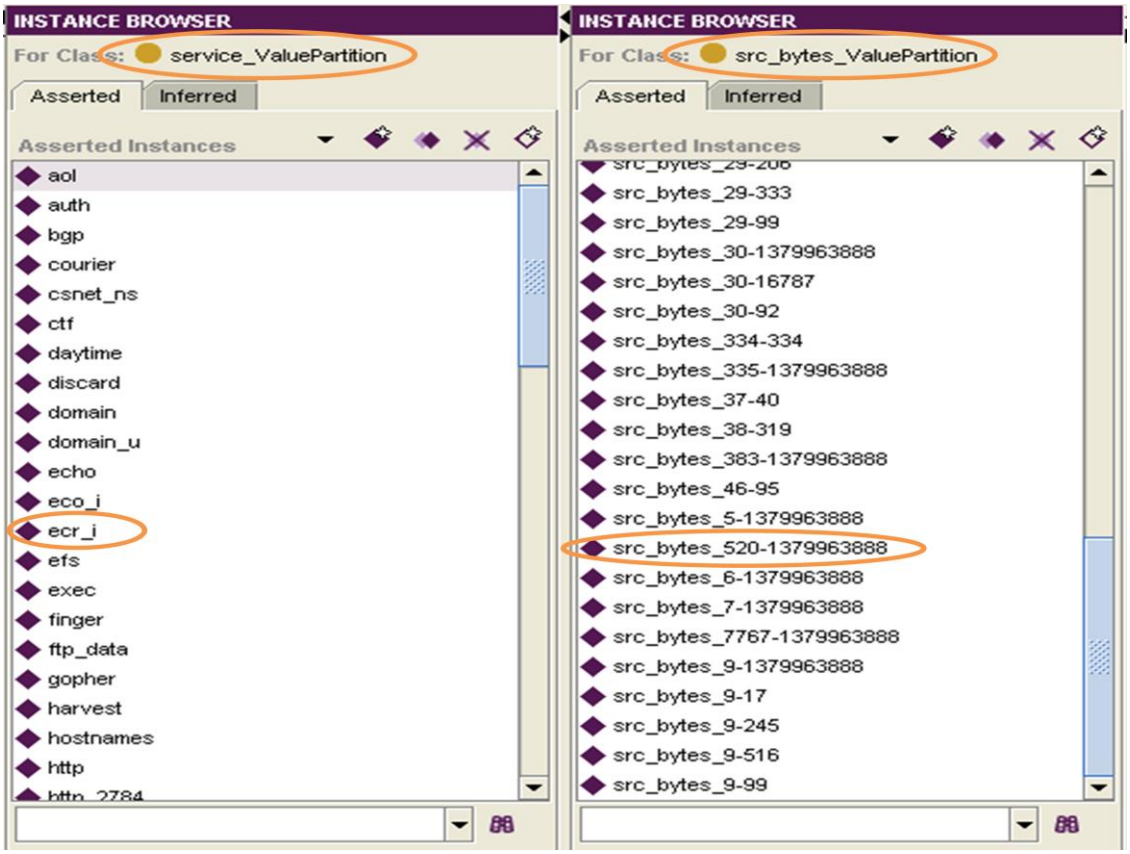
[۱] *Datatype Property*

و انتهای بازه می‌باشند. به عنوان مثال برای کلاس *src_bytes_ValuePartition* دو ویژگی نوع داده‌ای *src_bytes_end* و *src_bytes_start* تعریف می‌شوند (شکل ۵- ۶).

تمام مقادیر مجاز برای کلاس *src_bytes_ValuePartition* و کلاس *service_ValuePartition* در شکل ۵- ۷ نمایش داده شده‌است. پس از افزودن تمام مقادیر مجاز به کلاس‌های *ValuePartition* با استفاده از شروط یا همان بخش مقدم قوانین فاز سوم طراحی آنتولوژی کامل شده و نوبت به آخرین مرحله‌ی ساخت آنتولوژی می‌رسد.



شکل ۵- ۶ : تعریف مقادیر ابتدایی و انتهایی برای هر بازه



شکل ۵- ۷ : مقادیر مجاز برای کلاس *src_bytes_ValuePartition* و کلاس *service_ValuePartition*

۵- ۳- ۴- مقداردهی به خصایص برای نمونه‌های تعریف شده در آنتولوژی

آخرین مرحله ویا فاز چهارم از طراحی آنتولوژی مربوط به تعیین محدودیت‌ها و مقادیر مجاز ویژگی‌ها و خصایص کلاس‌های آنتولوژی می‌باشد. تنها تعریف کلاس‌ها برای داشتن آنتولوژی نمی‌تواند کافی باشد. برای اینکه آنتولوژی بتواند اهداف مورد نظر از طراحی را برآورده کند و پاسخ مناسب برای سوال‌ها و پرس و جوها داشته باشد، می‌بایست خصایص و ویژگی‌های کلاس‌های آن تعیین شود و محدودیت‌های آن‌ها مشخص باشد. در هر یک از ارتباطات شبکه، می‌بایست از نوعی پروتکل جهت هماهنگی ارتباطات استفاده شود و یا در ارتباطاتی که در شبکه برقرار می‌شود، طرفین از سرویس‌های خاصی می‌توانند استفاده کنند. هنگامی که حمله یا نفوذی در شبکه رخ می‌دهد، این نفوذها یا حمله‌ها نیز از این قاعده مستثنی نیستند. همانطور که پیشتر اشاره شد، با استفاده از نتایج حاصل از داده‌کاوی مشاهده شد که به عنوان مثال از ۲۶۴۶ رکورد اتصال مربوط به کلاس حملات *Smurf* تمام آنها دارای سرویس *ecr_i* و تعداد بایت‌های داده‌ای فرستاده شده از مبدا

به مقصد[۱]بیشتر از ۵۲۰ بایت می‌باشد و یا اینکه در کلاس حملات *Land* حتماً می‌بایست مقدار خصیصه‌ی *land* برابر با یک باشد.

اینگونه اطلاعات در طراحی آنتولوژی ما بسیار مهم بوده و به واسطه‌ی اینگونه اطلاعات می‌توان محدودیت‌ها و قیود مورد نیاز را برای اعضای آنتولوژی در نظر گرفت. به عنوان مثال در آنتولوژی "حملات کامپیوتری" می-بایست این نکته در نظر گرفته شود که نوع سرویس مورد استفاده‌ی تمامی اعضای کلاس *Smurf* از نوع *ecr_i* باشد و تعداد بایت‌های داده‌ای فرستاده شده از مبدا به مقصد بیشتر از ۵۲۰ بایت باشد و یا در رابطه با کلاس حملات *Land* تمامی اعضا تنها مجاز به داشتن خصوصیت *land* برابر با یک هستند.

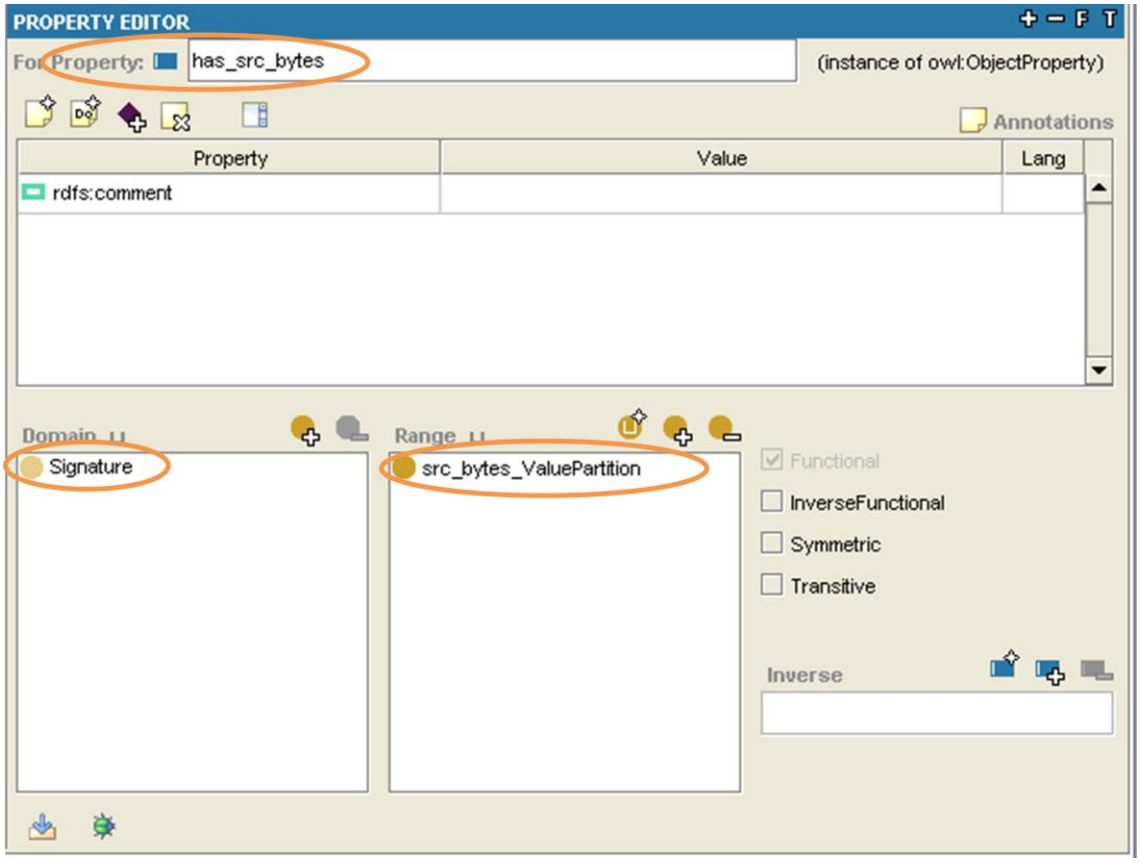
برای ایجاد این قیود و محدودیت‌ها از ویژگی شئی[۲]استفاده می‌شود. برای این کار برای خصیصه‌ی *src_bytes* ویژگی شئی *has_src_bytes* با دامنه‌ی *Signature* و برد *src_bytes_ValuePartition* از نوع تابعی[۳]تعریف می‌شود (شکل ۵- ۸)، این کار برای تمام خصیصه‌های ۴۱ گانه انجام می‌گیرد.

---

[۱] *src_bytes*

[۲] *Object Property*

[۳] *Functional*



شکل ۵- ۸ : تعریف ویژگی شئی *has_src_bytes*

در آخرین مرحله نیز با انتخاب کلاس مربوطه قیود مورد نظر برای آن تنظیم می‌شود. این مرحله از کار در مقایسه با بخش‌های قبلی بسیار وقت‌گیر بوده و احتیاج به دقت زیادی دارد. برای مثال قانون زیر را در نظر بگیرید:

(dst_host_srv_count <= 2) and (rerror_rate >= 0.25) and (service = private) and (dst_host_diff_srv_rate <= 0.04) and (count <= 8) => class=portsweep

برای این قانون قیود و محدودیت‌های زیر به آنتولوژی افزوده می‌شوند:

portsweep:

∃ has_dst_host_srv_count {dst_host_srv_count_0-2}

∃ has_rerror_rate {rerror_rate_0.25-1}

∃ has_service {private}

∃ has_dst_host_diff_srv_rate {dst_host_diff_srv_rate_0-0.04}

∃ has_count {count_0-8}

این عمل برای تمام کلاس‌ها انجام می‌گیرد. به واسطه‌ی قیود و محدودیت‌های تعیین شده برای هر یک از کلاس‌ها، در آنتولوژی "حملات کامپیوتری"، می‌توان روابط و ارتباط‌های کلاس‌های مختلف حملات را در آنتولوژی پیشنهادی بیان کرد. لازم به ذکر است که بیشتر قیود و محدودیت‌های تعیین شده در آنتولوژی پیشنهادی بر اساس اطلاعات و ویژگی‌هایی طراحی شده، که برگرفته از داده‌کاوی بر روی مجموعه‌ی داده‌های *NSL-KDD* هستند.

۵- ۳- ۵- بررسی صحت و درستی آنتولوژی طراحی شده

برای بررسی صحت و درستی آنتولوژی "حملات کامپیوتری" طراحی شده در این پایان‌نامه از دو روش استفاده شده است. ابتدا صحت و درستی آنتولوژی از نظر منطقی بررسی شده است. برای این کار از نرم‌افزار استنتاج‌گر *Racer* استفاده شده است*[Rac09]*. به کمک این نرم‌افزار می‌توان سازگاری[۱] کلاس‌های مختلف تعریف شده در آنتولوژی را بررسی کرد. به عنوان مثال ممکن است دو کلاس مختلف در آنتولوژی پیشنهادی به صورت جدا از هم[۲] تعریف شده باشند، مثالاً دو کلاس "نرمال" و "حمله" باید به صورت جدا از هم تعریف شوند چراکه یک وضعیت رخ داده در شبکه نمی‌تواند به طور همزمان هم در دسته‌ی نرمال قرار بگیرد و هم در دسته‌ی حمله. بنابراین اینگونه کلاس‌ها نمی‌توانند دارای نمونه‌های مشترک باشند اما ممکن است در یکی از مراحل طراحی آنتولوژی کلاسی تعریف شده باشد و یا ویژگی‌ها و محدودیت‌هایی که روی این کلاس‌ها در نظر گرفته شده طوری پیش رفته باشد که کلاس‌های جدا از هم بتوانند نمونه‌های مشترک با هم داشته باشند، که این امر در تناقض با تعریف اولیه‌ی آن کلاس‌ها که به صورت جدا از هم بوده می‌باشد. از آنجایی‌که در یک آنتولوژی ممکن است تعداد زیادی کلاس و ویژگی تعریف شده باشد، احتمال رخ دادن چنین اشتباهاتی زیاد است. بررسی عدم سازگاری‌ها کار پیچیده و زمانبری است اما در حال حاضر نرم‌افزارهای مختلفی برای این کار وجود دارد که یکی از رایج‌ترین آن‌ها نرم‌افزار *Racer* می‌باشد. البته این نرم‌افزار تنها موارد ناسازگاری رخ داده در آنتولوژی را پیدا کرده و بر طرف کردن این ناسازگاری‌ها و اصلاح آنتولوژی بر عهدهٔ خود طراح می‌باشد.

---

[۱] *Consistency*

[۲] *Disjoint*

لازم به ذکر است که کار طراحی آنتولوژی یک عمل تکرار شونده بوده و به مرور زمان و با انجام تست‌های مختلف در سیستم‌های واقعی می‌توان به یک آنتولوژی کامل دست پیدا کرد. انتظار این است که با تکامل آنتولوژی "حملات کامپیوتری" طراحی شده در این پایان‌نامه میزان خطای آن کاهش یابد. پس از انجام تست‌ها و بازبینی‌های فراوان بر روی آنتولوژی طراحی شده، آنتولوژی "حملات کامپیوتری" برای استفاده در سیستم تشخیص نفوذ پیشنهادی آماده شده و فایل *OWL* آن قابل استفاده در پیاده‌سازی این سیستم می‌باشد.

۵-۴- خلاصه

در این فصل ابتدا مروری بر نحوه طراحی و ایجاد آنتولوژی‌ها آورده شده و مراحل عملی ایجاد و ساخت آنتولوژی به تفصیل شرح داده شده است. در ادامه نیز نحوه‌ی طراحی آنتولوژی "حملات کامپیوتری" و چگونگی استخراج این آنتولوژی به تفصیل بیان شده است. در ادامه‌ی فاز طراحی آنتولوژی پیشنهادی در این پایان‌نامه و برای بدست آوردن روابط بین حملات و استخراج ویژگی‌های آن‌ها از روی مجموعه داده‌های *NSL-KDD*، تکنیک داده‌کاوی *RIPPER* مورد استفاده قرار گرفته است که در واقع یک روش یادگیری قانون می‌باشد. قوانین مستخراج از داده‌کاوی دربرگیرنده‌ی خصوصیات و ویژگی‌های هرکدام از کلاس‌های حملات بوده و با استفاده از این قوانین مقادیر مجاز برای ویژگی‌های یک حمله بدست می‌آید. پس از طراحی آنتولوژی "حملات کامپیوتری" در محیط *Protégé* فایل *OWL* تولید شده برای دستیابی به سایر اهداف مورد نظر در این پایان‌نامه قابل استفاده می‌باشد.

فصل ششم

«پیاده‌سازی و بررسی نتایج»

۶-۱- مقدمه

در این فصل یک سیستم تشخیص نفوذ مبتنی بر وب‌معنایی ارائه شده است. در این سیستم از آنتولوژی "حملات کامپیوتری" برای انجام عمل تشخیص نفوذ استفاده می‌شود. در مدل پیشنهادی در این پایان‌نامه یک سیستم چند عاملی در نظر گرفته شده است که دارای تعدادی عامل تشخیص نفوذ ثابت و متحرک مختلف و یک عامل مرکزی بوده که برای تشخیص یک نفوذ به صورت بهینه با یکدیگر همکاری می‌نمایند. امروزه استفاده از عامل‌ها به صورت قابل ملاحظه‌ای در ساخت سیستم‌های کامپیوتری افزایش یافته است. عامل به عنوان یک واحد پردازشی جدید مطرح شده و ساخت نرم افزارها بر اساس عامل رو به گسترش است. در این فصل ابتدا سیستم پیشنهادی *DIDMO*[۱] معرفی شده و نحوه‌ی عملکرد هر یک از عامل‌های موجود و تعاملات آن‌ها به تفصیل شرح داده خواهد شد و در ادامه نیز نحوه‌ی استنتاج عامل مرکزی با استفاده از یک تکنیک شباهت معنایی از روی آنتولوژی "حملات کامپیوتری" شرح داده خواهد شد. در انتهای فصل نیز آزمایشات و نتایج ارزیابی سیستم *DIDMO* بیان شده است.

۶-۲- تکنولوژی عامل‌ها

به علت گستردگی محدوده استفاده از عامل‌ها، تعاریف گوناگونی برای یک عامل ذکر شده است. در یک تعریف، عامل عبارت است از "یک سیستم نرم افزاری که در محیط قرار می‌گیرد و قادر به انجام اعمال انعطاف پذیر و مستقل در محیط، به منظور رسیدن به اهداف در نظر گرفته شده برای آن است". بدین ترتیب می‌توان یک عامل را موجودیتی در نظر گرفت که دارای هدف خاصی است و برای رسیدن به آن هدف، بر اساس شرایط مختلف تصمیم‌گیری می‌کند و می‌تواند رفتارهای مختلفی را از خود بروز دهد. "عامل متحرک" نیز یک برنامه‌ی کامپیوتری است که به صورت مستقل در برابر کاربر عمل می‌کند و از طریق یک سیستم توزیع شده به جایگاه‌های مختلف حرکت می‌کند. این نوع عامل‌ها از سه بخش کد، داده و وضعیت اجرایی تشکیل شده‌اند که بین گره‌های شبکه حرکت می‌کنند و به صورت مستقل در این گره‌ها فعالیت کرده و داده‌های لازم را جمع‌آوری می‌کنند *[Abd84]*.

---

[۱] *Distributed Intrusion Detection Using Mined Ontology*

استفاده نامناسب از عامل‌ها در سیستم‌های نرم‌افزاری، به عنوان نکته‌ی قابل توجهی مطرح است که باعث بروز مشکلات در سیستم‌های تولید شده می‌گردد. حال این سوال مطرح است که چگونه می‌توان تشخیص داد استفاده از عامل‌ها برای تولید چه نوع سیستم‌های نرم افزاری مناسب است؟ یک سیستم باید در بر گیرنده‌ی یک یا چند ویژگی از ویژگی‌های زیر باشد تا استفاده از تکنولوژی عامل در تولید آن مناسب باشد:

- کنترل توزیع شده
- ارتباط‌های پیچیده میان اجزا
- اهداف همزمان و گاه متضاد
- رفتارهای مستقل برای اجزای سیستم
- نیاز به انعطاف‌پذیری و انطباق

یک سیستم تشخیص نفوذ توزیع شده مسئول محافظت محیط کل شبکه در برابر نفوذ است. این امر نیاز به اطلاعات کامل وضعیت سیستم و مانیتور کردن قسمت‌های مختلف و ارتباط بین آن‌ها دارد و تکنولوژی عامل نقش حیاتی در این زمینه ایفا می‌کند. شبکه یک ساختار برای سیستم‌های توزیع شده می‌باشد بنابراین عامل یک انتخاب طبیعی برای این دیدگاه است. بدست آوردن اطلاعات از قسمت‌های مختلف شبکه، پردازش آن‌ها و پاسخ به درخواست آن‌ها بصورت مستقل می‌تواند از طریق عامل‌ها بدست آید.

۶-۳- طراحی محیط مبتنی بر عامل

برای استفاده از آنتولوژی طراحی شده‌ی "حملات کامپیوتری" و تست آن سعی شد تا شرایط یک شبکه واقعی برای آن شبیه‌سازی شود. بنابراین یک محیط چند عاملی برای رسیدن به این هدف در نظر گرفته شد. استفاده از محیط مبتنی بر عامل کمک می‌کند تا از قابلیت‌ها و رفتارهای عامل‌ها در برقراری ارتباط با یکدیگر و تعامل با یکدیگر جهت شبیه‌سازی بهتر محیط شبکه برای تست سیستم پیشنهادی استفاده شود. برای طراحی محیط مبتنی بر عامل از زبان برنامه‌نویسی *Java* به همراه کتابخانه‌ی *JADE*[۱] استفاده شده است.

[۱] *Java Agent DEvelopment Framework*

۶- ۳- ۱- بستر پیاده سازی عامل‌ها

در این تحقیق برای مدل‌سازی محیط مبتنی بر عامل در محیط برنامه‌نویسی جاوا از بسته‌ی *JADE* استفاده شده است. *JADE* یک بسته‌ی نرم‌افزاری نوشته شده به زبان جاواست که به صورت یک کتابخانه در اختیار برنامه‌نویس قرار می‌گیرد تا بتواند عامل‌های خود را به راحتی به زبان جاوا پیاده‌سازی کند*[Bel06]* . این کتابخانه به صورت متن‌باز بوده و به سرعت رو به گسترش می‌باشد. همراه با این کتابخانه، کتابخانه‌های دیگری نیز بوجود آمده‌اند که آن‌ها نیز از این بسته برای تولید استفاده کرده‌اند و هر کدام به نوع خاصی از عامل‌ها پرداخته‌اند و ابزارهای بسیار متنوعی جهت تولید و توسعه عامل‌ها در اختیار برنامه‌نویس قرار می‌دهند. ابزار *JADE* از روش*RMI*[۱] جهت ارتباطات استفاده می‌کند و از ویژگی‌های مهم این ابزار این است که لزومی ندارد برنامه‌نویس مسائل همزمانی متغیرها و توابع را شخصاً کنترل کند و این عملیات به صورت خودکار در سیستم اعمال می‌شوند.

۶- ۳- ۲- معرفی سیستم پیشنهادی

۶- ۳- ۲- ۱- معماری *DIDMO*

در تحقیق حاضر در طراحی سیستم پیشنهادی *DIDMO* مسائلی مانند انعطاف‌پذیری، مقیاس‌پذیری، مستقل بودن از پلتفرم و قابلیت اطمینان در نظر گرفته شده است. مولفه‌های *DIDMO* عبارتند از: *SA*، *MA*، *CA*، *Ontology*، *AA*، *MAD*، *DM*، *MB*، *MABB* و *SABB*. معماری *DIDMO* در شکل ۶- ۱ نشان داده شده است و در ادامه ویژگی‌های هر یک از مولفه‌های آن تشریح شده و به طرز کار سیستم و طریقه‌ی تعاملات این عامل‌ها پرداخته می‌شود.

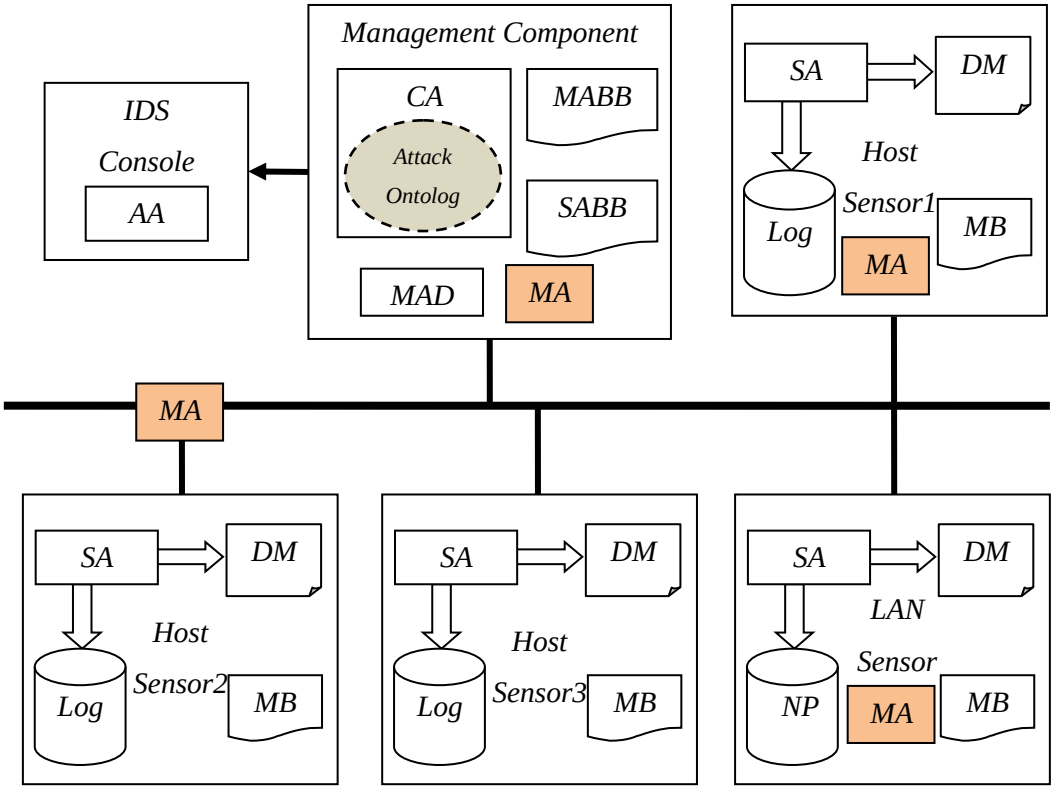
عامل ایستا (*SA*[۲]): نقش *SA* در معماری پیشنهادی یک مانیتور کننده یا حسگر می‌باشد و در مولفه‌های نصب شده بر روی تمام میزبان‌های تحت نظارت و مؤلفه‌های موجود بر روی حسگرهای شبکه یک *SA* قرار دارد که در ادامه به نقش آن پرداخته می‌شود.

[۱] *Remote Method Invocation*

[۲] *Static Agent*

شکل ۶- ۱: معماری *DIDMO*

عامل سیار (*MA*): این عامل‌ها وظیفه‌ی حرکت بین عامل‌های ایستا، دریافت اطلاعات مربوط به اتصالات، حذف اطلاعات اضافی، ادغام آن‌ها و تحویل این اتصالات به عامل مرکزی را به عهده دارند.



شکل ۶- ۱ : معماری *DIDMO*

مولفه‌ی مدیریت (*MC*)[۲]: این مؤلفه شامل آنتولوژی حملات در قالب یک فایل *OWL* بوده و با کمک یک عامل مرکزی استدلال و استنتاج بر روی این آنتولوژی را انجام داده، وضعیت اتصالات دریافتی از عامل‌های سیار را تشخیص داده و در صورت نفوذی بودن یک اتصال، هشدار مربوطه را به عامل هشدار تحویل می‌دهد.

عامل مرکزی (*CA*)[۳]: عامل اصلی در سیستم *DIDMO* می‌باشد که مجهز به آنتولوژی "حملات کامپیوتری" طراحی شده در این پایان‌نامه بوده و در واقع نقش نهایی تشخیص نفوذ را بر عهده دارد.

عامل هشدار (*AA*): این عامل هشدارهای دریافتی از مؤلفه‌ی مدیریت را به میز فرمانی[۲]که توسط مدیر امنیت شبکه مشاهده می‌شود می‌فرستد.

---

[۱] *Mobile Agent*

[۲] *Management Component*

[۳] *Central Agent*

گسیل کننده‌ی عامل‌های سیار [۳](MAD): نقش اصلی *MAD* اعزام عامل‌های سیار (برای جمع آوری داده-ها) به میزبان‌هایی که قبلاً هشدار اتصال مشکوک داشته‌اند می‌باشد.

تابلوی اعلانات عامل‌های ایستا [۴](SABB): نقش این مؤلفه در سیستم پیشنهادی تبادل اطلاعات بین عامل‌های ایستا و عامل مرکزی بوده و در واقع بخشی از مؤلفه‌ی مدیریت می‌باشد.

تابلوی اعلانات عامل‌های سیار [۵](MABB): نقش این مؤلفه در سیستم پیشنهادی تبادل اطلاعات بین عامل‌های سیار و عامل مرکزی بوده و در واقع بخشی از مؤلفه‌ی مدیریت می‌باشد.

تابلوی پیغام [۶](MB): این تابلو برای تبادل اطلاعات بین *SA*ها بوده و در تمام میزبان‌های تحت نظارت و حسگرهای شبکه موجود می‌باشد.

مدل داده‌ای [۷](DM): مدل داده‌ای نیز مانند تابلوی پیغام بخشی از مولفه‌ی نصب شده بر روی تمام میزبان-های تحت نظارت و حسگرها بوده و به تشخیص مشکوک بودن یک اتصال کمک کرده و باعث افزایش کارایی سیستم می‌شود.

۶- ۳- ۲- ۲- تعاملات عامل‌ها و تشخیص نفوذ

همان‌طور که گفته شد در معماری پیشنهادی *SA* نقش یک حسگر را به عهده داشته و در تمام مولفه‌های نصب شده بر روی میزبان‌های تحت نظارت و مؤلفه‌های موجود بر روی حسگرهای شبکه یک *SA* تعبیه شده است. مؤلفه‌های نصب شده بر روی میزبان‌های تحت نظارت با مشاهده‌ی فایل‌های ثبت وقایع[۸]میزبان‌ها که شامل *Application Logs*، *System Calls* و *System Status* بوده سعی در بیرون کشیدن ویژگی‌های مبتنی

---

[۱] *Alerting Agent*
[۲] *Console*
[۳] *Mobile Agent Dispatcher*
[۴] *Static Agent Bulletin Board*
[۵] *Mobile Agent Bulletin Board*
[۶] *Message Board*
[۷] *Data Model*
[۸] *Log Files*

بر میزبان یک اتصال می‌نماید. در هر بخش[1]از شبکه نیز یک مولفه حسگر شبکه وجود دارد که شامل یک *SA* بوده که نقش بیرون کشیدن خصوصیات مبتنی بر شبکه‌ی یک اتصال (خصوصیات ترافیکی و ذاتی یک اتصال) بر اساس محتویات بسته‌های عبوری در شبکه را برعهده دارد.

عامل‌های *SA* پس از بدست آوردن این ویژگی‌ها برای یک اتصال آن‌ها را با مدل داده‌ای خود مقایسه و در صورت مطابقت، این اتصال به‌عنوان اتصال مشکوک تشخیص داده می‌شود و *SA* پیغام هشداری برای *MAD* ارسال می‌نماید. این پیغام شامل شماره‌ی شناسایی اتصال مربوطه، آدرس میزبان و مجموع ویژگی‌های مشکوک اتصال بوده و نقش اطلاع رسانی به *MAD* برای ادامه‌ی روند تشخیص را بر عهده دارد.

برای جلوگیری از بمباران مؤلفه‌ی مدیریت توسط تمام ویژگی‌های یک اتصال بایستی تنها ویژگی‌های لازم و ضروری را به مؤلفه‌ی مدیریت تحویل داده تا این مؤلفه سریعتر و کارامدتر عمل نماید. برخی مواقع ممکن است که برای تشخیص یک نفوذ تنها ۲ یا ۳ خصیصه از ۴۱ خصیصه‌ی یک اتصال مورد نیاز باشد و با ارسال تمام ۴۱ خصیصه به مؤلفه‌ی مدیریت سربار محاسباتی زیادی بر روی آن اعمال شود. برای رسیدن به این هدف ابتدا عامل مرکزی پیغام‌های دریافتی از *SABB* را برداشته و به دنبال شباهت‌های معنایی بین آنتولوژی و خصیصه‌های موجود در این پیام‌ها می‌گردد. پس از یافتن بیشترین شباهت‌ها مجموعه‌ی خصیصه‌های مورد نیاز را برای تصمیم‌گیری نهایی در رابطه با وضعیت یک اتصال به همراه شماره‌ی شناسایی اتصال و آدرس میزبان در *MABB* قرار می‌دهد. عامل *MAD* نیز با توجه به اطلاعات موجود در *MABB* یک عامل سیار (*MA*) به میزبان مربوط گسیل[2]می‌کند.

این *MA* تمام خصیصه‌های مورد نیاز مبتنی بر میزبان و خصیصه‌های مورد نیاز مبتنی بر شبکه (خصیصه‌های ذاتی و خصیصه‌های ترافیکی) را از *SA*های موجود در میزبان‌های تحت نظارت و حسگرهای شبکه بدست آورده و پس از تجمع این خصوصیات و ایجاد اتصال موردنظر با خصیصه‌های درخواست شده توسط مؤلفه‌ی مدیریت، این اتصال را به مؤلفه‌ی مدیریت تحویل می‌دهد.

---

[1] *Segment*

[2] *Dispatch*

در این مرحله مؤلفه‌ی مدیریت به دنبال پیدا کردن شباهت‌های معنایی بین این اتصال و آنتولوژی حملات می‌گردد و پس از پیدا کردن بیشترین شباهت، نوع اتصال را بر اساس این شباهت تعیین و در صورت نفوذی بودن آن، هشدار مناسب را به عامل هشدار *(AA)* تحویل می‌دهد. عامل هشدار نیز هشدارهای دریافتی از عامل مرکزی را به میز فرمانی که توسط مدیر امنیتی شبکه مشاهده می‌شود ارسال می‌کند. علاوه بر این *AA* مسئول پیشگیری از نمایش هشدارهای مشابه در یک بازه‌ی زمانی معین بوده و اطلاعات مربوط به این هشدارها را نیز برای تحلیل‌های احتمالی آینده نگهداری می‌نماید.

۶- ۳- ۲- ۳- جزئیات بیشتر پیاده سازی

همان‌طور که گفته شد در تمام میزبان‌های تحت نظارت و یا در ناظرهای شبکه که به بررسی جزئیات بسته-های عبوری از شبکه می‌پردازند، بایستی یک عامل ایستا وجود داشته باشد. در واقع این عامل‌ها نقش مانیتور کردن تک‌تک میزبان‌ها و کل شبکه را به عهده داشته و هنگام تشخیص یک رفتار مشکوک بر اساس مدل داده‌ای پیغام مربوطه را به *MAD* ارسال می‌نمایند. در این حالت برای جلوگیری از ارسال پیام مشابه مربوط به یک اتصال مشکوک توسط چندین عامل ایستا بایستی شماره‌ی اتصال و مهر زمانی تشخیص آن در *MB* عامل‌های ایستای دیگر قرار بگیرد و پس از مشخص شدن وضعیت یک اتصال و با رعایت یک زمان اطمینان تمام پیغام‌های مربوط به آن اتصال از *MB* و*BB*های تمام مؤلفه‌ها حذف‌شود.

در سیستم پیشنهادی بر مقیاس‌پذیری سیستم تأکید زیادی شده است که بدین منظور می‌بایست بار محاسباتی عامل مرکزی کمترین مقدار ممکن باشد و بهمین‌دلیل باید از بمباران مؤلفه‌ی مدیریت توسط تمام اتصالات رخداده در شبکه به همراه خصیصه‌های ۴۱گانه‌ی آن‌ها جلوگیری کرده و از یک روش سرکشی[۱] استفاده شود. یعنی به جای اینکه یک عامل ایستا تمام اتصالات رخداده در شبکه را همراه با تمام خصوصیات آن‌ها به عامل مرکزی موجود در مؤلفه‌ی مدیریت بفرستد، تنها اتصالات مشکوک را همراه با خصوصیات درخواست شده توسط عامل مرکزی به عامل مرکزی تحویل دهد. از این‌رو عامل‌های ایستا به یک مدل داده‌ای مجهز شده که این مدل نقش بسزایی در این امر ایفا می‌کند و باعث می‌شود که برخی از اتصالات غیر

[۱] *Polling*

مشکوک در نظر گرفته شده و *SA*ها از ارسال خصیصه‌های آن‌ها به عامل مرکزی خودداری نمایند. این کار باعث کاهش حجم تعاملات و افزایش کارایی عامل مرکزی می‌شود.

مدل داده‌ای موجود در عامل‌های ایستا از داده‌کاوی‌های انجام شده بر روی مجموعه‌ی آزمایشی *NSL-KDD* تولید شده و در واقع ترکیبی از تمام حالت‌های ممکن برای هر یک از خصوصیات ۴۱گانه اتصالات نفوذی می‌باشد. بدین صورت که برای یک خصوصیت خاص مقادیر رخداده در تمام نفوذهای صورت گرفته بدست آمده و با هم ادغام می‌شود و عامل‌های ایستا به این مدل داده‌ای مجهز می‌شوند. عامل‌های ایستا تنها در مواقعی وضعیت یک اتصال را مشکوک قلمداد می‌نمایند که حداقل یکی از ویژگی‌های مبتنی بر میزبان (در عامل‌های ایستای مؤلفه‌های میزبان) و یا مبتنی بر شبکه (در عامل‌های ایستای مؤلفه‌های شبکه) یکی از مقادیر موجود در مدل داده‌ای را داشته باشد. با این کار بار محاسباتی بر روی عامل مرکزی کاهش یافته و بخشی از وظیفه‌ی آن بین عامل‌های ایستا توزیع می‌شود.

۶- ۳- ۲- ۴- مدل داده‌ای موجود در عامل‌های ایستا

در این بخش برای تولید مدل داده‌ای موجود در عامل‌های ایستا از نتایج داده‌کاوی استفاده شده است. با بررسی‌های آماری بر روی قوانین حاصل از اجرای الگوریتم داده‌کاوی *RIPPER* بر روی مجموعه داده‌های آزمایشی *NSL-KDD* مشاهده می‌شود که برخی از خصیصه‌های اتصال هیچگونه تاثیری در استخراج قوانین نداشته و در برخی از خصیصه‌های دیگر تنها مقادیر خاصی از آن‌ها در قوانین مشاهده شده است. در ادامه خصیصه‌هایی که در تولید قوانین نقشی نداشته اند مشاهده می‌شود:

is_guest_login, is_host_login, num_access_files, num_outbound_cmds, srv_rerror_rate, su_attempted, urgent

به زبان ساده‌تر می‌توان گفت هیچکدام از این هفت خصیصه جزء بخش مقدم[۱]هیچ قانونی نبوده و در کارایی روش ارائه شده هیچگونه تاثیری نداشته و جزء خصایص زائد می‌باشند. با حذف این خصیصه‌ها از یک اتصال به‌جای ۴۱ خصیصه در اتصالات ۳۴ خصیصه خواهیم داشت که این کاهش خصیصه به افزایش سرعت تشخیص و کاهش حجم داده‌های ارسالی بین عامل‌های مختلف کمک می‌کند.

---

[۱] *Antecedent*

با انجام تجزیه و تحلیل بیشتر بر روی قوانین حاصل از داده‌کاوی دریافت می‌شود که در برخی از خصیصه‌های چند مقداری، مقدار این خصیصه حالت بولی[۱]دارد یعنی تمامی مقادیر این خصیصه را می‌توان به دو گروه با مقادیر صفر و مقادیر غیر صفر تبدیل نمود که عملاً تمام مقادیر غیر صفر هیچ تفاوتی با یکدیگر ندارند. لیست خصایصی که دارای این ویژگی می‌باشند همراه با ویژگی‌های آن‌ها در جدول ۶- ۱ نشان داده شده است و پیشنهادی نیز برای جایگذاری این خصیصه‌ها ارائه شده است. احساس می‌شود که این جایگذاری خصیصه‌ها در مجموعه‌ی آزمایشی *KDD* به کاهش پیچیدگی‌های زائد این مجوعه کمک شایانی نماید.

جدول ۶- ۱ : لیست خصایص قابل اصلاح

نام خصیصه	مفهوم کنونی	مقادیر فعلی	مفهوم جدید پیشنهادی	مقادیر جدید
num_compromised	تعداد وضعیت‌های به خطر افتاده	*0-7479*	آیا وضعیت خطر افتاده وجود دارد؟	*0-1*
num_file_creations	تعداد عملیات تولید فایل	*0-43*	آیا عملیات تولید فایل صورت پذیرفته است؟	*0-1*
num_root	تعداد دسترسی‌ها به *ROOT*	*0-7468*	آیا دسترسی به *ROOT* صورت گرفته؟	*0-1*
root_shell	تعداد دستورات پوسته	*0-2*	آیا دستورات پوسته اجرا شده اند؟	*0-1*

سومین مساله‌ای که در ادمه‌ی بررسی رویت شد و به تولید مدل داده‌ای کمک زیادی نمود این بود که برای یک خصیصه‌ی خاص تنها برخی از مقادیر ممکن برای آن در قوانین آموخته شده به‌کار رفته است و سایر مقادیر آن بی‌اهمیت بوده‌اند (این مقادیر هیچ نقشی در تصمیم گیری وضعیت یک اتصال ندارند). با بهره‌گیری از این نگرش می‌توان به سادگی مشکوک یا غیر مشکوک بودن یک اتصال را تعیین نمود. اتصالات مشکوک اتصالاتی هستند که ممکن است جزء یکی از کلاس‌های نفوذ باشند که تصمیم‌گیری نهایی درباره وضعیت این اتصالات در عامل مرکزی انجام می‌پذیرد. ولی اتصالات غیر مشکوک بدون شک جزء کلاس‌های نرمال بوده و ضرورتی به ارسال این اتصال‌ها به عامل مرکزی وجود ندارد. برای تشخیص مشکوک بودن یک اتصال کافی است یک مدل داده‌ای ساده شامل تمام حالات ممکن برای هر یک از خصایص یک اتصال را در اختیار داشت، با یک مقایسه‌ی ساده می‌توان مشکوک بودن یک اتصال را تشخیص داد. یک اتصال زمانی مشکوک می‌باشد که حداقل یکی از خصایصش یکی از مقادیر ممکن مربوط به آن خصیصه در مدل داده‌ای را داشته باشد، و زمانی غیر مشکوک خواهد بود که هیچکدام از خصیصه‌هایش یکی از مقادیر ممکن مربوط به آن خصیصه در

[۱] *Boolian*

مدل داده‌ای را نداشته باشد. در جدول ۶- ۲ بازه‌های موجود و مورد نیاز در مدل داده‌ای برای برخی از خصیصه‌ها نمایش داده شده است.

در توضیح مبسوط جدول ۶- ۲ به عنوان نمونه در بخش مربوط به بازه‌های مشکوک خصیصه‌ی *land* مقدار *{1}* مشاهده می‌شود که حاکی از این است که فیلد *land* یک اتصال تنها زمانی باعث می‌شود که اتصال مشکوک شناخته شود که مقدار این خصیصه برابر با یک باشد. در غیر اینصورت این خصیصه نادیده گرفته می‌شود. و یا برای خصیصه‌ی *rerror_rate* مقادیر دو بازه‌ی *{0 to 0.14}* و *{0.25 to 1}* تعیین کننده هستند و مقادیری که در این بازه‌ها قرار ندارند نقش تعیین کننده‌ای در تصمیم گیری نداشته و نادیده گرفته می‌شوند.

جدول ۶- ۲ : مدل داده‌ای موجود در عامل‌های ایستا برای برخی خصیصه‌ها

عنوان خصیصه	نوع داده ای خصیصه	بیشترین مقدار خصیصه	اجتماع بازه‌های مشکوک
diff_srv_rate	اعشاری	*1*	*{0 to 0.2} , {0.36 to 1}*
dst_host_srv_rerror_rate	اعشاری	*1*	*{0.05 to 1}*
land	صحیح	*1*	*{1}*
num_failed_logins	صحیح	*5*	*{1 to 5}*
num_shells	صحیح	*2*	*{1 to 2}*
rerror_rate	اعشاری	*1*	*{0 to 0.14},{0.25 to 1}*
serror_rate	اعشاری	*1*	*{0 to 0.5}*
srv_diff_host_rate	اعشاری	*1*	*{1}*
srv_serror_rate	اعشاری	*1*	*{0.67 to 1}*
wrong_fragment	صحیح	*1*	*{1 to 3}*

۶- ۳- ۲- ۵- نحوه‌ی عملکرد عامل مرکزی

برای پیاده‌سازی اموری که مربوط به ارتباط با آنتولوژی، استخراج اطلاعات از آن و... می‌باشد، از بسته‌ی نرم-افزاری *Jena* استفاده شده است*[Jen09]*. *Jena* یک چارچوب به زبان *java* است که مناسب برای ایجاد برنامه‌های کاربردی مبتنی بر وب‌معنایی می‌باشد. بسته‌ی نرم افزاری *Jena* محیط برنامه‌نویسی برای *RDF*، *RDFS*، *OWL* و *SPARQL* بوده و شامل یک موتور استنتاج مبتنی بر قانون نیز می‌باشد.

عامل مرکزی حساس‌ترین بخش سیستم طراحی شده در این پایان‌نامه می‌باشد. این عامل مجهز به آنتولوژی "حملات کامپیوتری" شده است و آن را به صورت یک فایل *OWL* در اختیار گرفته است. عامل مرکزی با

دریافت یک گزارش از هر یک از عامل‌های سیار، موقعیت وضعیت گزارش شده را در آنتولوژی "حملات کامپیوتری" بررسی می‌کند که این عمل را به کمک زبان پرس‌وجو *SPARQL* که یک زبان پرس‌وجو برای *RDF* می‌باشد، انجام می‌دهد. عامل مرکزی به کمک این‌گونه پرس‌وجوها می‌تواند موقعیتِ وضعیت گزارش شده را در آنتولوژی "حملات کامپیوتری" استخراج کند. مثلاً در این پرس‌وجوها ویژگی‌هایی نظیر نوع پروتکل استفاده شده، نوع سرویس استفاده شده، وضعیت پرچم‌ها اعم از نرمال یا خطا بودن، تعداد بایت اطلاعاتی که برای مقصد ارسال شده و.... با وضعیت‌ها و ویژگی‌های تعریف شده در آنتولوژی مطابقت داده شده و به این صورت، وضعیت اتصال گزارش شده، توسط عامل مرکزی آنتولوژی "حملات کامپیوتری" تعیین می‌شود. به عبارت دیگر کلاسی از آنتولوژی که کاملاً با وضعیت گزارش داده‌شده مطابقت داشته باشد به عنوان کلاس این اتصال برگردانده می‌شود. یکی از معایب این روش این است که در صورتی که وضعیت جدیدی در شبکه رخ دهد که تفاوت بسیار اندکی با یکی از وضعیت‌های مورد استفاده در ساخت آنتولوژی داشته باشد، این سیستم قادر به‌طبقه بندی آن نمی‌باشد. در این تحقیق برای رفع این مشکل از یک تکنیک ساده برای یافتن شباهت‌های معنایی استفاده شده است.

C) شباهت معنایی در تشخیص نفوذ

در اینجا از یک تکنیک ابتکاری بسیار ساده برای یافتن میزان شباهت معنایی بین اتصال بدست آمده از یک وضعیت شبکه و کلاس‌های آنتولوژی حملات استفاده می‌شود. این تکنیک به‌جای تولید جواب قطعی[۱] یک جواب غیر قطعی[۲] تولید می‌نماید. یعنی اگر یک اتصال با هیچ یک از کلاس‌های آنتولوژی حملات مطابقت نداشته باشد، گره‌ای که بیشترین شباهت را با این اتصال دارد به عنوان کلاس مورد نظر برای طبقه‌بندی این اتصال انتخاب می‌نماید. همچنین میزان این شباهت را نیز در قالب یک عدد اعشاری مثبت کوچک‌تر از یک بیان می‌نماید. بیشترین مقدار شباهت یک بوده و زمانی اتفاق می‌افتد که یک کلاس از "آنتولوژی حملات" با اتصال مورد آزمایش کاملاً مطابقت داشته باشد.

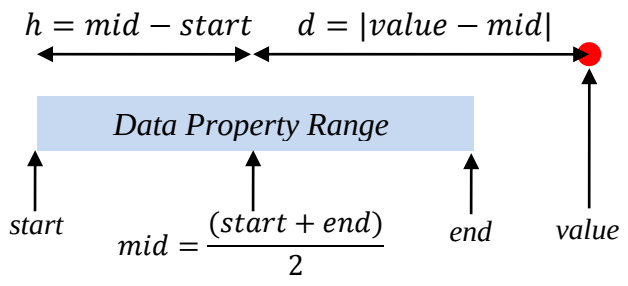
برای پیاده‌سازی این تکنیک در نظر گرفتن چند نکته به‌عنوان پیش‌فرض و یادآوری ضروری به نظر می‌رسد:

---

[۱] *Crisp*

[۲] *Fuzzy*

- رفتار خصیصه‌های یک اتصال با تغییر یکنواخت مقادیر عددی آن‌ها تغییر اندکی می‌کند و در واقع تابعی از میزان این تغییرات می‌باشد. مثلاً برای خصیصه‌ی *num_fail_logins* که تعداد *login*های ناموفق را بیان می‌کند، میزان شباهت بین مقادیر یک و دو بیشتر از میزان شباهت دو مقدار یک و پنج می‌باشد. و یا برای خصیصه‌ی *serror_rate* که درصد اتصالاتی را که خطای *SYN* داشته‌اند بیان می‌کند، شباهت بین دو مقدار *0.20* و *0.21* بیشتر از شباهت بین مقادیر *0.20* و *0.90* می‌باشد.
- رفتارهایی که در شبکه تفاوت اندکی با یکدیگر دارند در اغلب خصیصه‌ها دارای مقادیر یکسانی بوده و تنها در چند خصیصه دارای مقادیر متفاوت ولی نزدیک به یکدیگر می‌باشند.
- همان طور که در بخش پنجم ذکر شداز آنجا که نرم‌افزار *Protégé* امکان تعریف بازه را برای ما فراهم نمی‌کند، بازه‌های مورد نظر توسط خود ما با ترفند ساده‌ای تعریف شد. یعنی برای تمام زیر-کلاس‌های عددی[۱] *ValuePartition*ها دو ویژگی نوع داده‌ای[۲] با نام‌های *start* و *end* تعریف شد که ابتدا و انتهای بازه‌ی مورد نظر را مشخص می‌نمودند. در هنگام پرس‌وجو از آنتولوژی با استفاده از زبان *SPARQL* در محیط جاوا کافی است مقادیر تک‌تک خصیصه‌های عددی اتصال مورد آزمایش از مقدار *start* مربوط به خصیصه‌ی مشابه در *ValuePartition* بزرگ‌تر و از مقدار *end* مربوط به آن خصیصه در *ValuePartition* کوچک‌تر باشد. در این‌حالت با برقراری این شرط گفته می‌شود که این اتصال جزء کلاس مربوط از آنتولوژی حملات می‌باشد.



شکل ۶- ۲: شباهت جزئی خصیصه‌های عددی

[۱] *Numerical*

[۲] *Data Property*

در تکنیک پیشنهادی در این تحقیق از فاصله‌ها استفاده شد (شکل ۶- ۲)، یعنی اگر مقدار یک خصیصه در بازه‌ی مربوطه از کلاس مورد نظر در آنتولوژی قرار بگیرد میزان شباهت جزئی[۱]این خصیصه با خصیصه‌ی مشابه در کلاس موجود در آنتولوژی برابر با یک و یا بیشترین مقدار ممکن خواهد بود ولی اگر مقدار این خصیصه در بازه‌ی مربوطه قرار نگیرد میزان این شباهت جزئی با فاصله‌ی مقدار این خصیصه از مرکز بازه رابطه‌ی معکوس دارد، یعنی هرچه از مرکز بازه دورتر باشد شباهت جزئی کمتری خواهد داشت. شباهت کلی[۲] یک اتصال با یک کلاس آنتولوژی نیز حاصل میانگین این شباهت‌های جزئی می‌باشد.

شباهت جزئی حداکثر برابر با یک خواهد بود و این حالت زمانی اتفاق می‌افتد که میزان *value* یک خصیصه در بازه‌ی نوع داده‌ای قرار بگیرد، یعنی $start <= value <= end$ و در غیر اینصورت با استفاده از نسبت نصف اندازه‌ی بازه و فاصله از مرکز بازه محاسبه می‌شود.

$$partial\ proportion = min(1, \frac{h}{d})$$

برای خصیصه‌های غیر عددی یا اسمی[۳]مانند *protocol_type_ValuePartition*، *service_ValuePartition* و *flag_ValuePartition* پیدا کردن معیار شباهت یا قرابت بین مقادیر مختلف یک خصیصه اندکی مشکل‌تر می‌باشد زیرا اولاً این خصیصه‌ها دارای مقادیر غیر عددی بوده و ثانیاً حتی اگر این خصیصه‌ها مقادیر عددی نیز داشتند (در این حالت برای هر مقدار یک عدد در نظر می‌گیرند)، بازهم امکان استفاده از معیار فاصله نبود. مثلاً خصیصه‌ی *protocol_type_ValuePartition* دارای سه مقدار *tcp*، *icmp* و *udp* می‌باشد که پیدا کردن قرابت بین آن‌ها کار دشواری است. در اینجا نیز قبل از بیان روش پیشنهادی یافتن مقدار شباهت، لازم است چند نکته که فرضیات ما برای این کار بوده‌اند ذکر شود:

- برای یکی از این خصیصه‌های اسمی سه نمونه‌ی خاص را در نظر گرفته مثلا سه مقدار *tcp*، *icmp* و *udp* را از خصیصه‌ی *protocol_type_ValuePartition* در نظر بگیرید. اگر در یک نوع حمله‌ی خاص از هر دو نوع پروتکل *tcp* و *icmp* استفاده شود ولی از پروتکل *udp* استفاده نشود می‌توان

[۱] *Partial Proportion*
[۲] *Total Proporation*
[۳] *Nominal*

گفت این موضوع باعث افزایش قرابت دو مقدار *tcp* و *icmp* شده و میزان قرابت پروتکل *udp* با در-نظر گرفتن این نوع حمله کم می‌شود.

- مجدداً فرض کنید در یک نوع حمله‌ی خاص امکان وجود این سه پروتکل وجود داشته، ولی مقدار فراوانی آن‌ها متفاوت باشد، مثلاً از تمام ۱۰۰۰ حمله‌ی شناخته شده از این نوع ۵۰۰ حمله دارای پروتکل *tcp*، ۴۵۰ حمله دارای پروتکل *icmp* و تنها ۵۰ حمله دارای پروتکل *udp* باشد. با درنظر گرفتن این موضوع در یک نوع حمله‌ی خاص، می‌توان گفت نزدیک بودن مقدار فراوانی آنها به یکدیگر باعث افزایش یا کاهش قرابت آنها می‌شود. مثلاً در اینجا این مقادیر فراوانی باعث افزایش میزان قرابت دو پروتکل *tcp* و *icmp* شده و میزان قرابت پروتکل *tcp* نیز با دو پروتکل دیگر کم می‌شود.
- اگر حمله‌ی جدیدی از لحاظ رفتاری بسیار شبیه به یک حمله‌ی شناخته شده باشد، بیشتر خصیصه‌های هم نام آن‌ها با یکدیگر شباهت کامل داشته و تنها در چند خصیصه‌ی هم‌نام تفاوت اندکی وجود خواهد داشت که باز هم میزان شباهت این چند خصیصه بسیار زیاد خواهد بود. این موضوع برای تمام خصیصه‌های اسمی و عددی صدق می‌نماید.

در این تحقیق برای بدست آوردن میزان قرابت بین تمام مقادیر خصیصه‌های اسمی هم‌نوع با یکدیگر یک الگوریتم بسیار ساده و جالب طراحی شد. در این الگوریتم تمام فرضیات بالا در نظر گرفته شد و با بررسی‌های آماری بر روی مجموعه‌ی آزمایشی *NSL-KDD* و مقادیر مجاز برای خصایص اسمی آن‌ها میزان قرابت بین تک‌تک مقادیر مجاز یک خصیصه‌ی اسمی بدست آمد. لازم به ذکر است که از تمام ۴۱ خصیصه‌ی داده‌های آزمایشی *KDD* تنها سه خصیصه‌ی *flag*، *service* و *protocol_type* دارای مقادیر اسمی می‌باشند. الگوریتم پیشنهادی در محیط *C++* پیاده‌سازی شد و نتایج بسیار جالب و قابل توجهی تولید نمود. خروجی این الگوریتم برای مجموعه داده‌های آزمایشی سه ماتریس دو بعدی بود که هرماتریس به یک خصیصه‌ی اسمی تعلق داشت و مقدار درایه موجود بر روی سطر *i*ام و ستون *j*ام میزان قرابت بین *i*امین و *j*امین مقدار این خصیصه را بیان می‌نمود که یک عدد اعشاری مثبت کوچک‌تر از یک خواهد بود. مقدار صفر عدم وجود قرابت و مقدار یک وجود قرابت کامل را بیان می‌نماید.

جدول ۶- ۳: قرابت موجود بین پروتکل‌های *icmp*، *tcp* و *udp*

	icmp	***tcp***	***Udp***
icmp	*1*	*0*	*0.81*
tcp	*0*	*1*	*0.16*
udp	*0.81*	*0.16*	*1*

جدول ۶- ۳ میزان قرابت را برای هر سه مقدار خصیصه‌ی *protocol_type* بیان می‌نماید. مشخصاً بایستی تمام درایه‌های موجود بر روی قطر اصلی این ماتریس برابر با یک و عناصر ماتریس نسبت به قطر اصلی قرینه باشند. لازم به تذکر است که میزان قرابت یک پروتکل با خودش یک می‌باشد. نتایج جدول نشان می‌دهد که میزان قرابت بین *icmp* و *udp* نسبتاً زیاد بوده ولی بین *icmp* و *tcp* قرابتی وجود ندارد.

در پایان پس از یافتن شباهت‌های جزئی بین مقدار خصایص یک اتصال و مقدار ویژگی‌های یک کلاس از آنتولوژی، شباهت کلی با محاسبه‌ی میانگین این شباهت‌های جزئی محاسبه شد.

$$Total_{\ Proporatin} = \frac{\sum Partial_{Proporation}}{Count(Partial_{Proporation})}$$

تنها موضوع باقی مانده برای بحث میزان آستانه[1]و حساسیت سیستم برای هشدار می‌باشد که برای شبکه‌های گوناگون متفاوت بوده و با توجه به سیاست‌های امنیتی مدیر و مسئول امنیت شبکه و بازخوردهای سیستم به صورت دستی تنظیم می‌شود.

۶- ۴- آزمایش‌ها و بررسی نتایج

در این قسمت سیستم تشخیص نفوذ مبتنی بر وب‌معنایی طراحی شده در این پایان‌نامه مورد ارزیابی قرارمی گیرد. هدف اصلی از انجام این پایان نامه ارائه یک سیستم تشخیص نفوذ مبتنی بر وب معنایی در سطح شبکه‌های کامپیوتری بوده است و در آن یک عامل مرکزی با توجه به اطلاعاتی که از سایر سیستم‌های موجود در شبکه بدست می‌آورد، می‌تواند وضعیت امن بودن و یا ناامن بودن شبکه را بررسی کند.

در این بخش سیستم تشخیص نفوذ پیشنهادی ارزیابی شده و نتایج حاصل با چند روش دیگر مقایسه می-شود. ارزیابی سیستم بر اساس معیارهای مطرح در زمینه تشخیص نفوذ انجام گرفته است. نتایج ارزیابی سیستم که در ادامه‌ی این فصل آمده است نشان می‌دهد که این سیستم تشخیص نفوذ می‌تواند در طراحی

[1] *Threshold*

سیستم‌های تشخیص نفوذ بسیار موثر باشد و ما معتقدیم که این سیستم می‌تواند در زمینه تشخیص نفوذ بسیار بهتر از نتایج بدست آمده در این فصل عمل کند هر چند که دلیل قانع کننده‌ای برای این مدعا نداریم.

۶-۴-۱- بررسی نتایج و ارزیابی سیستم

داده‌های *NSL-KDD* شامل زیر مجموعه‌ای است که به آن زیر مجموعه تست یا آزمایش گفته می‌شود این زیر مجموعه شامل ۱۲۵۹۷۳ رکورد اتصال می‌باشد که هر کدام یکی از برچسب‌های کلاس‌های ۲۲ گانه حملات و یا کلاس داده‌های نرمال زده شده است. در این مجموعه داده حمله‌هایی وجود دارند که هیچ الگوی مشابهی برای آن‌ها در مجموعه داده‌های آموزش وجود ندارد و به آن‌ها حمله‌های جدید اطلاق می‌شود. در ادامه این فصل به ارزیابی سیستم تشخیص نفوذ ارائه شده در این پایان‌نامه بر اساس این مجموعه داده پرداخته شده است. برای آزمایش کارایی سیستم از میان مجموعه داده‌های تست *NSL-KDD*، ده نمونه داده-ی تست هر یک شامل ۱۲۰۰۰ رکورد به‌طور تصادفی انتخاب کرده و کارایی سیستم پیشنهادی با استفاده از این مجموعه بررسی می‌شود.

جدول ۶- ۴: نتایج طبقه بندی مجموعه داده‌های تست براساس معیارهای ارزیابی

مجموعه داده‌ی تست	TP[1]Rate	FP[2]Rate	ROC[3]Area	CCI[4]	ICI[5]	TNI[6]
سری شماره ۱	0.993	0.005	0.995	11912	82	12000
سری شماره ۲	0.994	0.005	0.996	11923	77	12000
سری شماره ۳	0.994	0.004	0.996	11931	69	12000
سری شماره ۴	0.994	0.004	0.996	11932	68	12000
سری شماره ۵	0.994	0.004	0.996	11929	71	12000
سری شماره ۶	0.994	0.004	0.995	11928	72	12000
سری شماره ۷	0.993	0.005	0.994	11913	87	12000
سری شماره ۸	0.995	0.004	0.996	11944	56	12000

[1] *True Posetive*
[2] *False Posetive*
[3] *Receiver Operating Characteristic*
[4] *Correctly Classified Instances*
[5] *Incorrectly Classified Instances*
[6] *Total Number of Instances*

سری شماره ۹	*0.994*	*0.004*	*0.9946*	*11931*	*69*	*12000*
سری شماره ۱۰	*0.993*	*0.005*	*0.995*	*11919*	*81*	*12000*
میانگین کل	***0.9938***	***0.0044***	***0.99536***	***119262***	***732***	***120000***

نتایج طبقه‌بندی مجموعه داده‌های تست براساس معیارهای تعریف شده در فصل دوم و هر یک از نسخه‌های تست سیستم *DIDMO* در جدول ۶- ۴ ذکر شده است. در این جدول نتایج قابل توجهی برای نسخه‌های حاصل از آموزش با سری داده‌های مختلف تست ارائه شده که حاکی از مؤثر بودن سیستم پیشنهادی و کارایی بالای آن می‌باشد. در جدول ۶- ۵ ماتریس برهم‌ریختگی ۲۳ کلاسه که در درک بهتر عملکرد طبقه بندی کننده موثر می‌باشد، برای مجموع تمام ده ساختار حاصل از آزمایشات ارائه شد.

جدول ۶- ۵ : بخشی از ماتریس بر هم ریختگی مربوط به طبقه بندی ۲۳ کلاسه اتصالات با کمک آنتولوژی حملات

	a	b	c	d	e	f	g	h	i	j	k
Normal = a	64015	14	13	8	32	0	0	0	1	1	43
PROBE-ipsweep = b	22	3375	20	0	3	0	0	0	0	0	0
PROBE-nmap = c	46	28	1332	1	2	0	0	0	0	0	2
PROBE-portsweep = d	26	3	7	2721	15	0	0	0	0	0	11
PROBE-satan = e	83	0	5	10	3371	0	0	0	0	0	10
DoS-pod = f	3	0	0	0	0	187	0	0	0	0	0
DoS-smurf = g	0	0	1	0	0	0	2522	0	0	0	0
DoS-teardrop = h	4	0	0	0	1	0	0	855	0	0	0
DoS-back = i	20	0	0	0	0	0	0	0	890	0	0
DoS-land = j	5	0	0	0	0	0	0	0	0	5	7
DoS-neptune = k	46	1	1	0	14	0	0	0	0	0	39177

جدول ۶- ۶: ماتریس بر هم ریختگی پنج کلاسه‌ی بدست آمده از ماتریس بر هم ریختگی جدول ۶- ۵

	Normal	PROBE	DoS	U2R	R2L
Normal	64015	67	45	2	42
PROBE	177	10893	23	0	3
DoS	78	18	43643	0	1
U2R	47	0	0	7	3
R2L	136	7	0	0	816

جدول ۶- ۷: ماتریس برهم ریختگی برای تست سیستم تشخیص نفوذ *DIDMO* بر روی مجموعه داده‌ای *NSL-KDD*

		Predicted					دقت
		Normal	PROBE	DoS	U2R	R2L	
Actual	**Normal**	64015	67	45	2	42	**0.99757**
	PROBE	177	10893	23	0	3	**0.98171**
	DoS	78	18	43643	0	1	**0.99778**
	U2R	47	0	0	7	3	**0.12281**
	R2L	136	7	0	0	816	**0.85089**
هشدار غلط		**0.0067**	**0.0083**	**0.0015**	**0.2222**	**0.0566**	**CPE=0.1129**

از آنجا که در اغلب روش‌های مشابه تنها به تشخیص کلاس‌های کلی حملات توجه شده و چهار کلاس مختلف حمله (*DoS*، *U2R*، *R2L*، *Probe*) و یک کلاس نرمال در نظر گرفته شده است ما برای مقایسه‌ی

روش خود با روش‌های دیگر ناچار به تولید ماتریس برهم ریختگی پنج کلاسه شدیم روش تولید این ماتریس در جدول ۶- ۶ نشان داده شده است.

لازم به ذکر است که با تولید این ماتریس تا حدودی از ارزش طبقه بندی خود در تحقیق کاسته شد زیرا تنها به تشخیص حملات در گروه‌های کلی پنج‌گانه اهمیت داده و خطاهای تشخیص بین زیرگروه‌ها را در نظر گرفته نشده است. با یک نگاه گذرا متوجه می‌شویم که عناصر قطر اصلی در ماتریس بر هم‌ریختگی نسبت به سایر عناصر دارای مقادیر بسیار بزرگی بوده که این امر حاکی از عملکرد خوب این روش می‌باشد.

نتایج جدول ۶- ۷ نشان می‌دهد که سیستم در کلاس *U2R* دارای میزان دقت بسیار پایینی است که می‌تواند عملکرد سیستم را تحت تاثیر قرار دهد. البته این اتفاق دور از انتظار نیست و علت آنرا می‌توان در پایین بودن تعداد نمونه‌های آموزشی در این کلاس حمله جستجو نمود. همچنین مشاهده می‌شود که تعداد زیادی از نمونه‌های دو کلاس *Normal* و *Probe* اشتباهاً جزء کلاس یک‌دیگر طبقه‌بندی شده اند که دلیل این موضوع را نیز می‌توان شباهت زیاد مدل رفتاری این دو کلاس به یکدیگر دانست. نتایج این جداول کارایی مناسب و درخور توجه سیستم *DIDMO* را نشان داده و مقدار *CPE* بدست آمده‌ی بسیار کم حاکی از عملکرد خوب این سیستم می‌باشد.

۶- ۴- ۲- مقایسه‌ی سیستم پیشنهادی با سیستم‌های مشابه

در ادامه برای ارزیابی هر چه بیشتر رویکرد فوق، سیستم *DIDMO* با چند روش یادگیری ماشین که نتایج آزمایشات خود را بر روی داده‌های *KDD* ارائه کرده‌اند مقایسه شد. جدول ۶- ۸ کارایی هر یک از این روش‌ها را بر اساس معیارهای ارزیابی تعریف شده در فصل دوم مقایسه می‌نماید. روش ارائه شده در بعضی از کلاس-های حمله کارایی بهتری نسبت به دیگر رقیبان ارائه کرده و مقدار *CPE* برابر با *0.1129* بیانگر توانایی این سیستم در تشخیص نفوذ می‌باشد. به راحتی از نتایج بدست آمده می‌توان استنباط نمود که سیستم *DIDMO* کارایی خوبی در تشخیص نفوذ در شبکه داشته و نرخ تشخیص و نرخ هشدارهای غلط قابل قبولی نیز ارائه می‌نماید. ما ادعا می‌کنیم که نتایج جدول در بعضی از سطرها می‌تواند غیر عادلانه باشد، برای مثال در *[Abd09]* تنها به تشخیص رفتارحملات *DoS* پرداخته شده است و سیستم تنها قادر به تشخیص *DoS* یا غیر *DoS* بودن رکوردهای اتصال بوده و هیچ اطلاعاتی را در مورد نوع رفتارهای غیر *DoS* ارائه نمی‌کند. بیان

حقیقت فوق دلیلی برای این مدعا نیست که چون سیستم ما یک طبقه‌بندی چهارکلاسه را برای حملات ارائه می‌کند روش مناسب‌تری می‌باشد، بلکه ما با بیان این مطلب قصد داریم توجه‌ی خواننده را به این نکته جلب کنیم که در روش فوق وقتی رکوردی به عنوان غیر *DoS* شناخته می‌شود کلاس آن هر چه که باشد به عنوان یک الگوی طبقه‌بندی شده‌ی درست در آن کلاس قرار می‌گیرد ولی در روش ارائه شده در این مطالعه بعضی از الگوها علیرغم تشخیص درست به‌عنوان غیر *DoS* ممکن است در کلاس نادرست طبقه‌بندی شوند. به عبارت دیگر در روش فوق تشخیص یک رفتار غیر *DoS* به معنی طبقه‌بندی درست است در حالی‌که در روش‌های دیگر که همانند روش ما عمل می‌کنند تشخیص حمله به معنای طبقه‌بندی درست نیست و امکان طبقه‌بندی اشتباه وجود دارد. اثبات این مدعا نرخ تشخیص حملات بالاتر در روش ارائه شده است که نشان می‌دهد روش ما در تشخیص حملات بسیار خوب عمل می‌نماید در حالیکه ممکن است در طبقه‌بندی کلاس دچار اشتباه شود. در پایان نتایج جدول دلیلی بر این مدعا هستند که روش *DIDMO* یک روش جدید و مناسب برای سیستم‌های تشخیص نفوذ می‌باشد که بر اساس ترکیب تکنیک‌های وب‌معنایی و چند روش داده‌کاوی عمل می‌کند.

جدول ۶- ۸: در صد نرخ طبقه بندی، نرخ تشخیص(*DTR*)، نرخ هشدارهای غلط(*FA*) و هزینه برای هر نمونه(*CPE*) در الگوریتم-های مختلف تشخیص نفوذ برای داده‌های تست مجموعه داده‌های *NSL-KDD*

CPE	***FA***	***DTR***	***R2L***	***U2R***	***DoS***	***Probe***	***Normal***	***الگوریتم***
0.1129	***0.2***	***99.2***	***85***	***12.2***	***99.7***	***98.1***	***99.7***	***DIDMO (our)***
0.015	*3*	*99.9*	*n/r*	*n/r*	*99.9*	*n/r*	*n/r*	*SWIDS[Abd09]*
0.1579	*1.9*	*95.3*	*31.5*	*14.1*	*99.5*	*84.1*	*98.2*	*ESC-IDS[Nad06]*
n/r	*3.5*	*94.4*	*12.4*	*76.3*	*99.7*	*86.8*	*96.5*	*RSS-DSS[Son05]*
0.2024	*n/r*	*n/r*	*31.2*	*93.6*	*96.7*	*99.2*	*97.4*	*Parzen-Window[Yeu02]*
0.2285	*n/r*	*n/r*	*9.6*	*29.8*	*97.3*	*88.7*	*n/r*	*Multi-Classifier[Sab03]*
0.2331	*0.6*	*91.8*	*8.4*	*13.2*	*97.1*	*83.3*	*99.5*	*Winner of KDD[Pfa00]*
0.2356	*0.6*	*91.5*	*7.3*	*11.8*	*97.5*	*84.5*	*99.4*	*Runner Up of KDD[Lev00]*
0.2371	*0.4*	*91.1*	*10.7*	*6.6*	*96.9*	*73.2*	*99.5*	*PNrule[Aga00]*

در رابطه با معیار *CPE* همانطور که پیش‌تر نیز اشاره شد، هرچه مقدار این معیار به عدد صفر نزدیک‌تر باشد نشان دهنده‌ی کارایی بهتر سیستم تشخیص نفوذ می‌باشد. مقدار *CPE* بدست آمده برای سیستم *DIDMO* برابر با *0.1129* بوده و تقریباً از سایر *CPE*های نشان داده شده در جدول ۶- ۸ کوچک‌تر می‌باشد. مقدار *CPE* در این جدول تنها از *CPE* بدست آمده در *SWIDS* بیشتر بوده که می‌توان علت این امر را در انتخاب

ماتریس هزینه‌ی متفاوت و دو کلاسی بودن آن روش دانست. روش ارائه شده در این تحقیق یک طبقه‌بند ۲۳ کلاسی بوده ولی در نتایج ارائه شده در این فصل پنج کلاس کلی در نظر گرفته شده یعنی علاوه بر تشخیص نفوذ سعی در تشخیص نوع آن نیز می‌شود. در ادامه با کمک گرفتن از ماتریس هزینه و برهم ریختگی استفاده شده در *[Abd09]* میزان *CPE* روش پیشنهادی بدست می‌آید.

جدول ۶- ۹: ماتریس برهم ریختگی برای مقایسه‌ی روش *DIDMO* با روش *SWIDS*

		Predicted		***دقت***
		DoS	***Not DoS***	
Actual	***DoS***	*43643*	*97*	***0.997782***
	Not Dos	*68*	*76192*	***0.999108***
هشدار غلط		***0.001556***	***0.001272***	***CPE=0.0218333***

در جدول ۶- ۹ مشاهده می‌شود که میزان پارامتر *CPE* محاسبه شده براساس ماتریس هزینه‌ی *[Abd09]* برابر با *0.021* می‌باشد که در مقایسه با مقدار بدست آورده شده در آن تحقیق *(0.015)* بیشتر بوده و این امر در نگاه اول نگران کننده به‌نظر می‌رسد ولی با کمی تامل به علت این موضوع پی می‌بریم. یکی از نکات اثبات شده در زمینه‌ی طبقه‌بندی این است که معمولا چند طبقه‌بند دوکلاسه طبقه‌بندی دقیق‌تر ولی کندتری نسبت به یک طبقه‌بند چندکلاسه ارائه می‌دهند*[Tso07]* . از این نکته می‌توان به این نتیجه رسید که هرچه تعداد کلاس‌های از پیش تعیین شده برای یک طبقه‌بندی کننده بیشتر باشد دقت طبقه‌بندی کمتر می‌شود. از آنجا که ما در این تحقیق در ساخت آنتولوژی خود از یک طبقه‌بند ۲۳ کلاسه استفاده کرده‌ایم و در *[Abd09]* از یک طبقه‌بند دو کلاسه استفاده شده است پس بطور ذاتی این دو روش قابل مقایسه نبوده و این میزان *CPE* دلیلی بر بدتر بودن کار ما نیست.

۶-۵- خلاصه

در این فصل ابتدا سیستم تشخیص نفوذ مبتنی بر وب معنایی پیشنهاد شده در این پایان‌نامه معرفی شده و هر یک از اجزای آن شرح داده شده است. در ادامه نحوه‌ی عملکرد عامل مرکزی و نحوه‌ی استفاده‌ی آن از آنتولوژی "حملات کامپیوتری" شرح داده شده، چگونگی انجام عمل پرس‌وجو و استنتاج از آنتولوژی "حملات کامپیوتری" نیز با کمک تکنیک‌های مشابهت معنایی مختصراً توضیح داده شده است. سپس به بررسی

عملکرد سیستم *DIDMO* پرداخته و آزمایش‌های مختلفی که بر روی آن انجام شده است، مورد ارزیابی قرار گرفته‌اند. در انتها نیز حاصل این ارزیابی‌ها در قالب نمودار و جداول نشان داده شده و نتایج حاصل از سیستم *DIDMO* با سایر روش‌های موجود مقایسه و ارزیابی شده است. سیستم تشخیص نفوذ پیشنهادی درصد نرخ طبقه‌بندی، نرخ تشخیص(*DTR*)، نرخ هشدارهای غلط(*FA*) و هزینه برای هر نمونه(*CPE*) قابل قبولی ارائه نموده و نشانگر کارایی تکنیک‌های وب‌معنایی و مناسب بودن این روش در حوزه‌ی تشخیص نفوذ می‌باشد.

فصل هفتم

« نتیجه گیری و کارهای آتی»

۷–۱– اهداف پایان‌نامه

در این فصل اهداف پایان نامه به طور مختصر بیان خواهد شد و نتایج بدست آمده مرور می‌شود. در انتها موضوعاتی که می‌تواند در ادامه کار این پایان‌نامه مورد بررسی بیشتر قرار بگیرد، مطرح خواهد شد.

به طور کلی در این پایان‌نامه یک روش تشخیص نفوذ توزیع‌شده‌ی مبتنی بر وب معنایی ارائه شده و برای تشخیص وضعیت‌های رخ داده در سیستم از آنتولوژی "حملات کامپیوتری" استفاده می‌شود. هر گزارشی که از وضعیت‌های رخ‌داده در شبکه برای عامل مرکزی ارسال شود، در آنتولوژی "حملات کامپیوتری" بررسی شده و نتیجه حاصل برای عامل هشدار ارسال می‌شود.

در سیستم پیشنهادی بر مقیاس‌پذیری سیستم توجه زیادی شده و می‌بایست بار محاسباتی عامل مرکزی کمترین مقدار ممکن باشد. برای نیل به این هدف بایستی از بمباران عامل مرکزی توسط تمام اتصالات رخداده در شبکه به همراه خصیصه‌های ۴۱گانه‌ی آنها جلوگیری کرده و از یک روش سرکشی[۱]استفاده شود. یعنی به جای اینکه یک عامل ایستا تمام اتصالات رخداده در شبکه را همراه با تمام خصوصیات آن‌ها به عامل مرکزی بفرستد، تنها اتصالات مشکوک را همراه با خصوصیات درخواست شده توسط عامل مرکزی به عامل مرکزی تحویل دهد. از این‌رو عامل‌های ایستا به یک مدل داده‌ای مجهز شده که این مدل نقش بسزایی در این امر ایفا می‌کند و باعث می‌شود که برخی از اتصالات با کمک مدل داده‌ای موجود در عامل‌های ایستا غیر-مشکوک در نظر گرفته شده و عامل‌های ایستا از ارسال خصیصه‌های آن‌ها به عامل مرکزی خودداری نمایند. این کار باعث کاهش حجم تعاملات و افزایش کارایی عامل مرکزی می‌شود.

برای تولید آنتولوژی کاویده شده، بررسی صحت آنتولوژی "حملات کامپیوتری" و تست سیستم *DIDMO* طراحی شده در این پایان‌نامه از مجموعه داده‌های *NSL-KDD* استفاده شده است. نتایج حاصل از آزمایش‌ها و تست‌های انجام شده، نشان از عملکرد خوب سیستم *DIDMO* می‌باشد. معیارهایی مانند: معیار *CPE* و نرخ تشخیص بدست آمده برای سیستم *DIDMO* گواهی بر کارایی خوب سیستم *DIDMO* می‌باشد. ممکن است تنها عیب سیستم *DIDMO* این باشد که سیستم در کلاس *U2R* دارای میزان دقت بسیار پایینی بوده که

---

[۱] *Polling*

می‌تواند عملکرد سیستم را تحت تاثیر قرار دهد. البته این اتفاق دور از انتظار نیست و علت آن‌را می‌توان در پایین بودن تعداد نمونه‌های آموزشی در این کلاس حمله جستجو نمود. اما موضوع قابل توجه نرخ بسیار بالای تشخیص بوده که از نقاط قوت این سیستم است.

در این تحقیق برای بدست آوردن ویژگی‌ها و خصوصیات دامنه‌ی حملات با استفاده از داده‌های آزمایشی *NSL-KDD* از الگوریتم داده‌کاوی *RIPPER* استفاده شده است. همچنین برای پیاده‌سازی آنتولوژی حملات از نرم‌افزار *Protégé* استفاده شده است. پیاده‌سازی سیستم *DIDMO* نیز در محیط برنامه نویسی *Java* و با بهره‌گیری از بسته‌های نرم‌افزاری *Jade* و *Jena* انجام گرفته است.

۷– ۲– تحقیقات بیشتر

تصور می‌شود تحقیق انجام شده در این پایان‌نامه گام جدیدی است که از مفاهیم وب معنایی و بخصوص از آنتولوژی مستخرج از داده‌کاوی در جهت بهبود سیستم‌های تشخیص نفوذ استفاده کرده است. امکان تکمیل و ادامه‌ی تحقیق حاضر از جهت‌های دیگری نیز میسر می‌باشد. اولین کار دارای اولویت توسعه و بهبود آنتولوژی حملات کامپیوتری می‌باشد به گونه‌ای که بتوان به‌سادگی بر اساس شباهت‌های موجود بین مقادیر یک خصیصه و استفاده از تکنیک‌های مختلف شباهت معنایی نفوذهای جدید و ناشناخته را نیز تشخیص داد.

از طرفی مدل داده‌ای استفاده شده در عامل‌های ایستا یا همان سنسورها بسیار ضعیف بوده و می‌توان با تقویت این مدل داده‌ای بر میزان تشخیص رکوردهای غیر مشکوک افزود، این کار باعث کاهش حجم داده‌های ارسالی بین عامل‌های ایستا و عامل مرکزی و افزایش مقیاس پذیری سیستم می‌شود.

از فعالیت‌های دیگری که در ادامه این پروژه می‌توان انجام داد توسعه آنتولوژی در جهتی است که عمل همکاری بین عامل‌ها توسعه پیدا کند. همچنین ادامه‌ی تحقیق بر روی عمل استنتاج بر روی آنتولوژی به گونه‌ای که بتوان تعامل بین عامل‌های موجود در محیط چند عامله را در جهت تشخیص حملات توزیع شده افزایش داد.

زمینه‌ی کاری دیگری که در طی فرایند انجام این تحقیق مشاهده شد، استفاده از تکنیک‌های مختلف داده-کاوی بر روی داده‌های آزمایشی موجود و بررسی آماری نتایج برای بهینه‌سازی تعداد خصایص رکوردها و مقدار هر خصیصه می‌باشد.

منابع

[Aba05] *M. S. Abadeh, J. Habibi, C. Lucas, "Intrusion detection using a fuzzy genetics-based learning algorithm", Journal of Network and Computer Applications, August 2005.*

[Aba80]

م. آبادی، س. جلیلی، "شناسایی فعالیت‌های نفوذی در سطح ترافیک شبکه با استفاده از روش یادگیری ماشین"، هفتمین کنفرانس سراسری انجمن کامپیوتر ایران، اسفند ۱۳۸۰.

[Abd07] *F. Abdoli, M. Kahani, "Using Attacks Ontology in Distributed Intrusion Detection System", International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE07), December, 2007.*

[Abd09] *F. Abdoli, M. Kahani, "Ontology-based distributed intrusion detection system", Computer Conference (CSICC09), 14th International CSI, Page(s): 65-70, 2009.*

[Abd84]

ا. عبداله زاده بارفروش، ب. معصومی، م. آیت الله زاده شیرازی، "مقدمه ای بر هوش مصنوعی توزیع شده (معرفی عامل و سیستم‌های چند عامله)"، انتشارات جلوه، ۱۳۸۴.

[Aga00] *R. Agarwal, M. V. Joshi, "PNrule: A New Framework for Learning classifier Models in Data Mining", Technical Report TR 00-015, Depratment of Computer Science, University of Minnesota, 2000.*

[Ana05] *T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, "Enabling attack behavior prediction in ubiquitous environments", Pervasive Services, ICPS '05, 2005.*

[And95] *D. Anderson, T. F Lunt, H. Javitz, A. Tamaru, A. Valdes, "Detecting Uniusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES)", Technical Report, SRI International, May 1995.*

[Axe00] *S. Axelsson, "Intrusion detection systems: A survey and taxonomy, Department of Computer Engineering", Chalmers University, Tech. Rep. 99-15, 2000.*

[Bac02] *R. Bace, P. Mell, "Intrusion detection system", NIST Special Publication on Intrusion Detection, 2002.*

[Bar01] *D. Barbarra, J. Couto, S. Jajodia, L. Popyach, "ADAM: Detecting Intrusion by Data Mining", Proceeding of the 2001 IEEE Workshop on Information Assurance and Security T1A3 1100 nited Staes Military Academy, West Point, Ny, June 2001.*

[Bel06] *F. Bellifemine, G. Caire, T. Trucco, G. Rimassa, "JADE Programmer's Guide", 2006.*

[Ber01] *T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web", The Scientific American. May 2001.*

[Blo01] *E. Bloedorn, "Data Mining for Network Intrusion Detection: How to Get Started", Technical paper, 2001.*

[Bor97] *W. Borst, "Construction of Engineering Ontologies for Knowledge Sharing and Reuse", Center for Telematics and Information Technology, University of Twente, 243 pages, 1997.*

[Bur04] *K. Burbeck, S. Nadjm-Tehrani, "Adwice-anomaly detection with real-time incremental clustering", In Proceedings of the 7th International Conference on Information Security and Cryptology, Seoul, Korea, 2004.*

[Chi01] *A. Chittur, "Model generation for an intrusion detection system using genetic algorithms", High School Honors Thesis, Ossining High School. In cooperation with Columbia Univ, 2001.*

[Coh95] *W. W. Cohen, "Fast effective rule induction", In ICML, page(s): 115-123, 1995.*

[Cro95] *M. Crosbie, E. H. Spafford, "Active defense of a computer system using autonomous agents", Technical Report CSD-TR-95-008, Purdue Univ., West Lafayette, IN, 15 February 1995.*

[Das02] *D. Dasgupta, F. Gonzalez, "An immunity-based technique to characterize intrusions in computer networks", IEEE Trans. Evol. Comput. 6 (3), 1081-1088, June 2002.*

[Deb99] *H. Debar, M. Dacier, A. Wespi, "Towards a taxonomy of intrusion detection systems", Computer Networks, 31(8):805–822, April 1999.*

[Den87] *D. E. Denning, "An Intrusion Detection Model", IEEE Transaction on Software Engineering, Vol. SE-13, No. 2, pp. 222-232, February 1987.*

[Dik00] *J. E. Dickerson, J. A. Dickerson, "Fuzzy Network Profiling for Intrusion Detection", Proceedings of NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society, Atlanta, July, 301-306, 2000.*

[Dik01] *J. E. Dickerson, J. Juslin, O. Koukousoula, J. A. Dickerson, "Fuzzy intrusion detection", IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference, Vancouver, British Columbia, Volume 3, 1506-1510, July, 2001.*

[Dow90] *C. Dowell, P. Ramstedt, "The Computer Watch Data Reduction Tool", In Proceedings of the 13th National Computer security Conference, pp. 99-108, Washington, DC, October 1990.*

[Fay96] *U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, "The KDD process of extracting useful knowledge from volumes of data", Communications of the ACM, 39(11):27–34, November 1996.*

[Gho99] *A. K. Ghosh, A. Schwartzbard, M. Schatz, "Learning program behavior profiles for intrusion detection", In Proc. 1st USENIX, 9-12 April, 1999.*

[Gol89] *D. E. Goldberg, E. David, "Genetic Algorithms in Search, Optimization & Machine Learning", Addison-Wesley, 1989.*

[Gom01] *J. Gomez, D. Dasgupta, "Evolving Fuzzy Classifiers for Intrusion Detection", Proceeding Of 2002 IEEE Workshop on Information Assurance, United States Military Academy, West Point NY, June 2001.*

[Gru93] *T. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", In the Proceedings of the Workshop on Formal Ontoloy. Padua, March, 1993.*

[Gua03] *Y. Guan, A. Ghorbani, N. Belacel, "Y-means: A Clustering Method for Intrusion Detection", Proceedings of Canadian Conference on Electrical and Computer Engineering. Montreal, Quebec, Canada. May 4-7, 2003.*

[Han01] *J. W. Haines, L. M. Rossey, R. P. Lippman, R. K. Cunningham, "Extending the darpa off-line intrusion detection evaluations", In DARPA Information Survivability Conference and Exposition II, volume 1, pages 77 – 88. IEEE, 2001.*

[Hof99] *S. A. Hofmeyr, S. Forrest, "Immunizing computer networks: Getting all the machines in your network to fight the hacker disease", In Proc. of the 1999 IEEE Symp. On Security and Privacy, Oakland, CA. IEEE Computer Society Press, 1999.*

[Ilg95] *K. Ilgun, R. A. Kemmerer, P. A. Porras, "State transition analysis: A rule-based intrusion detection approach", IEEE Transactions on Software Engineering, 21 (3), 181--199, 1995.*

[Jan93] *J.-S. R. Jang, "ANFIS: Adaptive-Network-based Fuzzy Inference Systems", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 3, pp. 665-685, May 1993.*

[Jen09] *Available in http://jena.sourceforge.net/*

[Ken99] *K. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems", Master's thesis, MIT, 1999.*

[KDD99] *KDD Cup 1999 Intrusion detection dataset: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.*

[Kum94] *S. Kumar, E. H. Spafford, "A pattern-matching model for misuse intrusion detection", In NIST (Ed.), Proceedings of the 17th national computer security conference (pp. 11-21), National Institute of Standards and Technology (NIST), Baltimore, MD, 1994.*

[Kum95] *S. Kumar, "Classification and detection of computer intrusions", Unpublished doctoral dissertation, Purdue University, West Lafayette, IN, 1995.*

[Lee99] *W. Lee, S. J. Stolfo, K. Mok, "A data mining framework for building intrusion detection models", Proceedings of IEEE Symposium on Security and Privacy, pp 120 –132, 1999.*

[Lev00] *I. Levin, "KDD-99 Classifier Learning Contest LLSoft's Results Overview", SIGKDD Explorations, ACM SIGKDD, 1(2) 67-75, 2000.*

[Lin99] *U. Lindqvist, P. A. Porras, "Detecting computer and network misuse through the production-based expert system toolset (PBEST)", In L. Gong & M. Reiter (Eds.), Proceedings of the 1999 IEEE symposium on security and privacy (pp. 146--161), IEEE Computer Socitey, Los Alamitos, CA, 1999.*

[Luo03] *G. Luo, X. L. Lu, J. Li, J. Zhang, "Madids: A novel distributed ids based on mobile agent", ACM SIGOPS Operating Systems Review, vol. 37, pp. 46-53, Jan. 2003.*

[Man05] *S. Mandujano, A. Galván, J. A. Nolazco, "An Ontology-based Multiagent Architecture for Outbound Intrusion Detection", 3rd ACS/IEEE International Conference on Computer Systems and Applications, AICCSA ´05, vol. 1, pp. 120-128, Cairo, Egypt, January 2005.*

[McH00] *J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory", ACM Trans. Information System Security 3 (4), 262294, 2000.*

[Mou95] *A. Mounji, B. L. Charlier, D. Zampuniéris, N. Habra, "Distributed audit trail analysis", In D. Balenson & R. Shirey (Eds.), Proceedings of the ISOC'95 symposium on network and distributed system security (pp. 102--112), IEEE Computer Society, Los Alamitos, CA, 1995.*

[Muk02] *S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines", in International Joint Conference onNeural Networks IJCNN02, vol. 2, pp. 1702–1707, Honolulu, HI USA, May 2002.*

[Muk03] *S. Mukkamala, A. H. Sung, "Identifying significant features for network forensic analysis using artificial intelligent techniques", International Journal of Digital Evidence 1 (4), 1-17, 2003.*

[Nad06] *A. Nadjaran-Toosi. , M. Kahani, R. Monsefi, "Network Intrusion Detection Based on Neuro-Fuzzy Classification", ICOCI2006 (Kuala Lumpur, Malaysia, June 6-8, 2006.*

[Nat01] *F. Natalya, D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology", Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.*

[Nig01] *P. Ning, S. Jajodia, X. S. Wang, "Abstraction-based intrusion in distributed environments", ACM Transactions on Information and Systems Security, 4(4):407 – 452, November 2001.*

[Pfa00] *B. Pfahringer, "Winning the KDD99 Classification Cup: Bagged Boosting", SIGKDD explorations, 1(2), 65-66, 2000.*

[Pro09] *Available in http://protege.stanford.edu*

[Rac09] *Available in http://www.sts.tu-harburg.de/~r.f.moeller/racer/*

[Ram04] *G. Ramachandran, D. Hart, "A P2P intrusion detection system based on mobile agents", in Proceedings of the 42nd annual southeast regional conference, pp. 185–190. ACM Press New York, NY, USA, Apr. 2004.*

[Ras01] *V. Raskin, C. Helpenmann, K. Triezenberg, S. Nirenburg, "Ontology in information security: a useful theoretical foundation and methodological tool", New Security Paradigms Workshop, ACM Press, pp. 53-59, Cloudcroft, NM, 2001.*

[Sab03] *M. R. Sabhnani, G. Serpen, "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context", Proceedings of International Conference on Machine Learning: Models, Technologies, and Applications, Las Vegas, Nevada, 209-215, 2003.*

[Sim04] *Y. F. Simon, M. Arradondo, "Mobile agents for computer intrusion detection", in Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory, pp. 517–521, IEEE, 2004.*

[Sod04] *A. S. Sodiya, "A new combined strategy for intrusion detection", PhD Thesis, Department of Mathematical Sciences, University of Agriculture, Abeokuta, Nigeria, 2004.*

[Son05] *D. Song, M. I. Heywood, A. N. Zincir-Heywood, "Training Genetic Programming on Half a Million Patterns: An Example from Anomaly Detection", IEEE Transactions on Evolutionary Computation, 2005.*

[Sta02] *S. Staniford, J. A. Hoagland, J. M. McAlerney, "Practical automated detection of stealthy portscans", Journal of Computer Security 10 (1-2), 105-136, 2002.*

[Tan06] *D. Taniar, J. W. Rahayu, "Web Semantics Ontology", Idea Group Publishing, 2006.*

[Tav09] *M. Tavallaee, E. Bagheri, W. Lu, A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set", Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.*

[Tso07] *S. H. Tsong, J. L. Tsung, J. L. Yuh, "A three-tier IDS via data mining approach", The special issue on Machine Learning and Robust Data Mining of Computational Statistics and Data Analysis, 2007.*

[Und03] *J. Undercoffer, A. Joshi, J. Pinkston, "Modeling Computer Attacks: An Ontology for Intrusion Detection", Springer, pp. 113–135, 2003.*

[Und04] *J. Undercoffer, A. Joshi, T. Finin, J Pinkston, "A target centric ontology for intrusion detection: using DAML+OIL to classify intrusive behaviors", Knowledge Engineering Review, Cambridge University Press, pp. 23-29, January, 2004.*

[Vap95] *V. N. Vapnik, "The Nature of Statistical Learning Theory", Springer-Verlag, New York, 1995.*

[Wan09] *Y. Wang, "Statistical Techniques for Network Security: Modern Statistically-Based Intrusion Detection and Protection", Information Science reference, Hershey, New York, 2009.*

[Yan04] *H. Yanxiang, C. Wei, Y. Min, P. Wenling, "Ontology Based Cooperative Intrusion Detection System", Network and Parallel Computing, 2004.*

[Yan05] *W. Yan, E. Hou, N. Ansari, "Extracting and querying network attack scenarios knowledge in IDS using PCTCG and alert semantic networks", IEEE International Conference 2005.*

[Yeu02] *D. Y. Yeung, C. Chow, "Parzen-window Network Intrusion Detectors", Sixteenth International Conference on Pattern Recognition, Quebec City, Canada, pp. 11-15, August 2002.*

[Zha01] *R. Zhang, D. Qian, C. Ba, W. Wu, X. Guo, "Multi-agent based intrusion detection architecture", in Proceedings of 2001 IEEE International Conference on Computer Networks and Mobile Computing, pp. 494–501, Oct. 2001.*

ضمائم

ضمیمه الف: مشخصات مجموعه داده‌ای *NSL-KDD*

Basic features of individual TCP connections	
Length (number of seconds) of the connection	*Duration*
Type of the protocol {ICMP (yes/no), TCP (yes/no), UDP (yes/no)}	*protocol_type*
Network service on the destination, HTTP (yes/no)	*Service*
Number of data bytes from source to destination	*src_bytes*
Number of data bytes from destination to source	*dst_bytes*
Normal or error status of the connection {REJ (yes/no), S0 (yes/no), SF (yes/no), RSTO or RSTOS0 or RSTR (yes/no)}	*Flag*
Connection from/to the same host/port (yes/no)	*Land*
Number of WRONG fragments	*Wrong_fragment*
Number of urgent packets	*Urgent*
Content features within a connection suggested by domain knowledge	
Number of HOT indicators	*Hot*
Number of failed login attempts	*num_failed_logins*
Login successfully (yes/no)	*Logged_ in*
Number of COMPROMISED conditions	*num_compromised*
Root shell s obtained	*root_shell*
SU ROOT command attempted (yes/no)	*su_attempted*
Number of ROOT accesses	*num_root*
Number of file creation operations	*num_file_creations*
Number of shell prompts	*num_shells*
Number of operations on access control files	*num_access_files*
Number of outbound commands in an ftp session	*num_outbound_cmds*
HOT login (yes/no)	*is_hot_login*
GUEST login (yes/no)	*is_guest_login*
Traffic features computed using a two-second time window	
Number of connections to the same host as the current connection in the past two seconds	*Count*
Percent of connections that have SYN errors	*Serror_rate*
Percent of connections that have REJ errors	*Rerror_rate*
Percent of connections to the same service	*same_srv_rate*
Percent of connections to different services	*diff_srv_rate*
Number of connections to the same service as the current connection in the past two seconds	*srv_count*
Percent of connections that have SYN errors	*srv_serror_rate*
Percent of connections that have REJ errors	*srv_rerror_rate*
Percent of connections to different hosts	*srv_diff_host_rate*
Destination	
Number of connections having the same destination host	*dst_host_count*
Number of connections having the same destination host and using the same service	*dst_host_srv_count*
Percent of connections having the same destination host and using the same service	*dst_host_same_srv_rate*
Percent of different services on the current host	*dst_host_diff_srv_rate*
Percent of connections to the current host having the same source port	*dst_host_same_src_port_rate*
Percent of connections to the same service coming from different hosts	*dst_host_srv_diff_host_rate*
Percent of connections to the current host that have an S0 error	*dst_host_serror_rate*
Percent of connections to the current host and specified service that have an S0 error	*dst_host_srv_serror_rate*
Percent of connections to the current host that have an RST error	*dst_host_rerror_rate*
Percent of connections to the current host and specified service that have an RST error	*dst_host_srv_rerror_rate*

جدول ۱: ویژگی‌های مربوط به داده‌های *NSL-KDD*

جدول ۲: حملات شناخته شده و جدید در داده‌های *NSL-KDD*

DOS	***Known***	*Back, land, Neptune, Pod, smurf, teardrop*
	Novel	*apache2, udpstorm, processtable, mailbomb*
Probe	***Known***	*ipsweep, satan, nmap, portsweep*
	Novel	*Saint, mscan*
R2L	***Known***	*ftp_write, guess_passwd, warezmaster, warezclient, imap, phf, spy, multihop*
	Novel	*named, xlock, sendmail, xsnoop, worm, snmpgetattack, snmpguess*
U2R	***Known***	*rootkit, loadmodule, buffer_overflow, perl*
	Novel	*xterm, p.s., sqlattack, httptunnel*

جدول ۳: تعداد نمونه‌های موجود در مجموعه داده‌های آزمایشی *NSL-KDD*

نوع دسته		**تعداد نمونه**
Normal		*67343*
Dos	*Back*	*956*
	land	*18*
	Neptune	*41214*
	Pod	*201*
	smurf	*2646*
	teardrop	*892*
	apache2	-
	udpstorm	-
	processtable	-
	mailbomb	-
Probe	*ipsweep*	*3599*
	satan	*3633*
	nmap	*1493*
	portsweep	*2931*
	Saint	-
	mscan	-
R2L	*ftp_write*	*8*
	guess_passwd	*53*
	warezmaster	*20*
	warezclient	*890*
	imap	*11*
	phf	*4*
	spy	-
	multihop	*6*
	named	-
	xlock	-
	sendmail	-
	xsnoop	-
	worm	-
	snmpgetattack	-
	snmpguess	-
U2R	*rootkit*	*10*
	loadmodule	*9*
	buffer_overflow	*30*
	perl	*3*
	xterm	-
	p.s	-
	sqlattack	-
	Httptunnel	-
Total		*125973*

ضمیمه ب: قوانین بدست آمده برای انواع مختلف حملات و رفتار نرمال با استفاده از روش طبقه‌بندی *RIPPER*

Attack Rules

1) *(num_shells >= 1) and (dst_host_same_src_port_rate <= 0.01) => class=perl (4.0/1.0)*
2) *(root_shell >= 1) and (src_bytes <= 51) and (service = http) => class=phf (4.0/0.0)*
3) *(num_file_creations >= 1) and (dst_host_srv_count <= 1) and (num_shells >= 1) => class=multihop (3.0/1.0)*
4) *(dst_host_count <= 2) and (service = login) => class=ftp_write (2.0/0.0)*
5) *(dst_host_count <= 2) and (service = ftp_data) and (dst_host_srv_count >= 84) and (dst_host_srv_count <= 85) => class=ftp_write (2.0/0.0)*
6) *(num_file_creations >= 1) and (src_bytes <= 116) and (src_bytes >= 104) => class=ftp_write (2.0/0.0)*
7) *(dst_host_srv_diff_host_rate >= 0.4) and (service = ftp_data) and (logged_in <= 0) => class=ftp_write (2.0/0.0)*
8) *(dst_host_count <= 6) and (diff_srv_rate >= 0.5) and (dst_bytes >= 2072) => class=loadmodule (4.0/1.0)*
9) *(service = telnet) and (dst_host_count <= 6) and (dst_bytes <= 1301) and (src_bytes >= 135) => class=loadmodule (5.0/1.0)*
10) *(num_file_creations >= 1) and (dst_host_same_srv_rate <= 0.02) and (dst_host_srv_count >= 4) => class=rootkit (2.0/0.0)*
11) *(service = imap4) and (dst_host_diff_srv_rate <= 0.01) => class=imap (14.0/3.0)*
12) *(land >= 1) => class=land (25.0/7.0)*
13) *(dst_bytes >= 1159100) and (service = ftp_data) and (duration <= 10) => class=warezmaster (16.0/0.0)*
14) *(root_shell >= 1) and (hot >= 3) and (dst_bytes <= 16771) => class=buffer_overflow (16.0/0.0)*
15) *(num_file_creations >= 1) and (dst_host_count <= 5) and (duration <= 290) and (dst_bytes >= 698) => class=buffer_overflow (5.0/0.0)*
16) *(service = ftp_data) and (dst_bytes >= 2072) and (logged_in >= 1) and (dst_host_srv_diff_host_rate <= 0.02) => class=buffer_overflow (8.0/0.0)*
17) *(num_failed_logins >= 1) and (hot >= 1) and (duration <= 60) => class=guess_passwd (52.0/1.0)*
18) *(wrong_fragment >= 1) and (protocol_type = icmp) => class=pod (198.0/0.0)*
19) *(service = tim_i) and (dst_host_count <= 2) => class=pod (4.0/1.0)*
20) *(src_bytes >= 334) and (src_bytes <= 334) and (service = ftp_data) => class=warezclient (425.0/1.0)*
21) *(duration <= 6) and (hot >= 28) => class=warezclient (273.0/0.0)*
22) *(duration >= 4) and (service = ftp_data) and (dst_host_same_srv_rate >= 0.91) => class=warezclient (104.0/1.0)*
23) *(hot >= 3) and (duration >= 5039) and (dst_bytes <= 2284) => class=warezclient (39.0/0.0)*
24) *(service = ftp_data) and (dst_host_same_src_port_rate >= 1) and (dst_host_srv_diff_host_rate >= 0.11) => class=warezclient (32.0/1.0)*
25) *(service = ftp_data) and (duration >= 4) and (dst_host_serror_rate >= 0.01) and (logged_in >= 1) => class=warezclient (7.0/1.0)*
26) *(service = ftp_data) and (dst_host_same_srv_rate >= 1) and (src_bytes <= 246) and (src_bytes >= 246) => class=warezclient (6.0/0.0)*

27) *(wrong_fragment >= 1) => class=teardrop (892.0/0.0)*
28) *(src_bytes >= 21048) and (service = http) => class=back (949.0/1.0)*
29) *(src_bytes >= 13140) and (service = http) => class=back (9.0/1.0)*
30) *(dst_host_same_src_port_rate >= 0.79) and (srv_diff_host_rate >= 1) and (dst_host_srv_diff_host_rate <= 0.33) and (dst_host_srv_diff_host_rate >= 0.25) and (srv_count >= 3) => class=nmap (918.0/0.0)*
31) *(flag = SH) and (hot <= 0) => class=nmap (265.0/0.0)*
32) *(dst_host_same_src_port_rate >= 0.79) and (service = private) and (src_bytes >= 207) and (dst_host_count <= 250) => class=nmap (207.0/0.0)*
33) *(service = eco_i) and (dst_host_srv_diff_host_rate <= 0.4) and (src_bytes <= 8) and (dst_host_srv_count >= 3) => class=nmap (49.0/0.0)*
34) *(dst_host_same_src_port_rate >= 0.11) and (service = private) and (src_bytes >= 100) and (src_bytes <= 100) => class=nmap (38.0/1.0)*
35) *(service = ecr_i) and (src_bytes >= 520) => class=smurf (2646.0/0.0)*
36) *(flag = RSTR) and (src_bytes <= 1) => class=portsweep (2178.0/1.0)*
37) *(diff_srv_rate >= 0.36) and (rerror_rate >= 1) and (dst_host_same_src_port_rate >= 0.01) => class=portsweep (482.0/0.0)*
38) *(flag = RSTOS0) => class=portsweep (99.0/0.0)*
39) *(dst_host_srv_count <= 1) and (dst_host_same_src_port_rate >= 0.04) and (src_bytes <= 0) and (dst_host_count >= 71) => class=portsweep (141.0/0.0)*
40) *(dst_host_rerror_rate >= 0.01) and (srv_serror_rate >= 0.67) and (dst_host_same_src_port_rate >= 0.14) and (count >= 35) => class=portsweep (10.0/0.0)*
41) *(dst_host_rerror_rate >= 0.01) and (service = eco_i) and (src_bytes <= 8) and (srv_count <= 2) => class=portsweep (4.0/0.0)*
42) *(dst_host_srv_count <= 2) and (rerror_rate >= 0.25) and (service = private) and (dst_host_diff_srv_rate <= 0.04) and (count <= 8) => class=portsweep (5.0/0.0)*
43) *(dst_host_rerror_rate >= 0.01) and (dst_host_srv_serror_rate >= 0.03) and (dst_host_same_src_port_rate >= 0.05) and (dst_host_count >= 73) => class=portsweep (5.0/0.0)*
44) *(src_bytes >= 16980502) and (duration >= 12672) => class=portsweep (2.0/0.0)*
45) *(dst_host_srv_diff_host_rate >= 0.25) and (service = eco_i) and (src_bytes <= 18) => class=ipsweep (3060.0/0.0)*
46) *(dst_host_diff_srv_rate >= 0.99) and (dst_host_count <= 72) and (dst_host_rerror_rate >= 0.57) => class=ipsweep (470.0/2.0)*
47) *(dst_host_same_src_port_rate >= 1) and (protocol_type = icmp) and (src_bytes <= 18) => class=ipsweep (70.0/13.0)*
48) *(dst_host_count <= 4) and (service = ftp_data) and (flag = REJ) => class=ipsweep (11.0/0.0)*
49) *(dst_host_diff_srv_rate >= 0.16) and (src_bytes <= 6) and (dst_host_count >= 199) and (count >= 4) => class=satan (2956.0/0.0)*
50) *(protocol_type = udp) and (src_bytes <= 5) and (dst_host_count >= 69) => class=satan (466.0/0.0)*
51) *(diff_srv_rate >= 1) and (count >= 4) and (dst_host_rerror_rate >= 0.01) => class=satan (114.0/0.0)*
52) *(dst_host_srv_count <= 6) and (count <= 8) and (src_bytes <= 20) and (src_bytes >= 20) => class=satan (29.0/1.0)*
53) *(diff_srv_rate >= 0.5) and (src_bytes <= 19) and (dst_host_srv_count <= 1) and (serror_rate <= 0) => class=satan (27.0/6.0)*
54) *(dst_host_diff_srv_rate >= 0.31) and (src_bytes <= 37) and (num_compromised >= 1) => class=satan (4.0/1.0)*
55) *(service = finger) and (duration >= 1) and (srv_count >= 2) => class=satan (16.0/0.0)*

56) *(dst_host_diff_srv_rate >= 0.12) and (dst_host_rerror_rate >= 0.03) and (dst_host_srv_count >= 124) and (dst_host_count >= 176) => class=satan (7.0/0.0)*
57) *(dst_host_srv_count <= 5) and (count <= 34) and (src_bytes <= 40) and (src_bytes >= 37) and (logged_in <= 0) and (dst_host_serror_rate <= 0) => class=satan (5.0/0.0)*
58) *(same_srv_rate <= 0.49) and (src_bytes <= 0) and (srv_count >= 3) => class=neptune (35617.0/1.0)*
59) *(dst_host_serror_rate >= 0.92) and (dst_host_srv_serror_rate >= 0.98) and (flag = S0) => class=neptune (4488.0/0.0)*
60) *(same_srv_rate <= 0.03) and (rerror_rate >= 1) => class=neptune (778.0/2.0)*
61) *(flag = S0) and (dst_host_srv_serror_rate >= 0.12) and (dst_host_serror_rate >= 0.18) => class=neptune (286.0/1.0)*
62) *(src_bytes <= 0) and (dst_host_same_srv_rate <= 0.46) and (count >= 4) and (diff_srv_rate <= 0.2) => class=neptune (42.0/0.0)*
63) *(dst_host_serror_rate >= 0.91) and (flag = S0) and (service = smtp) => class=neptune (3.0/0.0)*
64) *(service = private) and (dst_host_rerror_rate >= 1) => class=neptune (4.0/0.0)*

Normal Rules

1) *(dst_bytes >= 4) and (hot <= 0) and (dst_host_srv_count >= 129) and (dst_host_rerror_rate <= 0) => class=normal (44185.0/2.0)*
2) *(src_bytes >= 29) and (src_bytes <= 333) and (rerror_rate <= 0.14) and (srv_count >= 2) and (dst_bytes >= 101) => class=normal (3502.0/2.0)*
3) *(srv_count <= 24) and (dst_host_same_srv_rate <= 0.81) and (hot <= 0) and (protocol_type = tcp) and (src_bytes >= 335) and (dst_host_diff_srv_rate <= 0.06) => class=normal (4538.0/4.0)*
4) *(count <= 15) and (dst_host_diff_srv_rate >= 0.03) and (src_bytes <= 205) and (src_bytes >= 104) => class=normal (2134.0/6.0)*
5) *(count <= 15) and (dst_host_same_src_port_rate <= 0.5) and (dst_host_srv_diff_host_rate >= 0.01) and (dst_host_same_srv_rate >= 0.53) and (service = http) => class=normal (2788.0/0.0)*
6) *(src_bytes >= 9) and (srv_count <= 2) and (dst_host_same_srv_rate <= 0.81) and (hot <= 25) and (dst_host_srv_count >= 5) and (src_bytes <= 99) and (dst_host_srv_count <= 67) and (rerror_rate <= 0) => class=normal (1411.0/0.0)*
7) *(src_bytes <= 333) and (dst_host_same_srv_rate <= 0.78) and (src_bytes >= 101) and (num_file_creations <= 0) and (dst_host_srv_serror_rate <= 0) => class=normal (1105.0/0.0)*
8) *(srv_count <= 25) and (dst_host_diff_srv_rate >= 0.03) and (protocol_type = tcp) and (dst_host_rerror_rate <= 0) and (src_bytes >= 335) and (duration <= 3) => class=normal (1765.0/0.0)*
9) *(src_bytes >= 9) and (srv_count <= 24) and (dst_host_diff_srv_rate >= 0.01) and (dst_host_count <= 254) and (dst_bytes >= 1) and (duration >= 8) => class=normal (854.0/8.0)*
10) *(count <= 24) and (src_bytes >= 29) and (src_bytes <= 99) and (dst_bytes <= 112) and (dst_host_count >= 13) and (dst_host_same_src_port_rate >= 0.02) => class=normal (1213.0/0.0)*
11) *(src_bytes >= 9) and (dst_host_diff_srv_rate >= 0.03) and (protocol_type = tcp) and (hot <= 0) and (dst_host_same_src_port_rate <= 0.31) and (count <= 1) and (dst_host_rerror_rate <= 0.04) and (dst_bytes >= 36) => class=normal (328.0/0.0)*
12) *(count <= 12) and (src_bytes <= 206) and (src_bytes >= 29) and (dst_host_srv_count >= 53) and (logged_in <= 0) => class=normal (230.0/0.0)*
13) *(count <= 24) and (dst_bytes >= 4) and (hot <= 0) and (src_bytes >= 7) and (dst_host_srv_count >= 3) and (dst_host_count <= 153) and (dst_host_serror_rate <= 0.39) and (same_srv_rate >= 0.33) => class=normal (524.0/0.0)*
14) *(count <= 19) and (srv_count <= 2) and (dst_host_diff_srv_rate <= 0.13) and (src_bytes <= 17) and (dst_host_serror_rate <= 0.94) and (dst_host_srv_count >= 4) and (protocol_type =*

tcp) and (dst_host_same_src_port_rate <= 0.88) and (count <= 1) and (dst_host_count <= 170) => class=normal (335.0/0.0)

15) *(count <= 2) and (dst_host_srv_count >= 3) and (src_bytes <= 960) and (dst_host_serror_rate <= 0.03) and (dst_host_same_src_port_rate <= 0.08) and (dst_host_same_srv_rate >= 0.04) and (hot <= 1) and (dst_host_rerror_rate <= 0.1) and (dst_host_srv_count >= 5) => class=normal (315.0/0.0)*
16) *(count <= 29) and (src_bytes >= 9) and (service = ftp_data) and (dst_host_srv_diff_host_rate <= 0.06) and (src_bytes <= 245) => class=normal (240.0/0.0)*
17) *(count <= 34) and (src_bytes >= 30) and (src_bytes <= 16787) and (dst_bytes >= 2492) and (root_shell <= 0) and (num_compromised <= 0) => class=normal (171.0/2.0)*
18) *(count <= 29) and (flag = REJ) and (dst_host_same_srv_rate >= 0.22) and (service = http) => class=normal (417.0/0.0)*
19) *(protocol_type = tcp) and (hot <= 0) and (dst_host_srv_diff_host_rate <= 0.06) and (src_bytes >= 335) and (duration <= 0) and (dst_host_srv_count <= 124) => class=normal (283.0/4.0)*
20) *(dst_host_srv_count <= 4) and (same_srv_rate <= 0.67) and (dst_host_rerror_rate <= 0) and (src_bytes >= 30) and (duration <= 1) => class=normal (74.0/0.0)*
21) *(dst_bytes >= 1) and (src_bytes <= 966) and (dst_host_rerror_rate <= 0.08) and (duration <= 52) and (src_bytes >= 231) => class=normal (108.0/0.0)*
22) *(src_bytes <= 95) and (src_bytes >= 46) and (duration <= 1) and (dst_bytes <= 1163) => class=normal (124.0/0.0)*
23) *(src_bytes >= 9) and (src_bytes <= 516) and (dst_bytes >= 1) and (hot <= 0) and (dst_host_same_srv_rate >= 0.61) and (dst_host_srv_count >= 3) => class=normal (26.0/0.0)*
24) *(count <= 5) and (dst_host_srv_diff_host_rate <= 0.1) and (dst_host_count <= 45) and (protocol_type = tcp) and (dst_host_srv_count >= 145) => class=normal (72.0/0.0)*
25) *(count <= 7) and (src_bytes >= 9) and (dst_host_srv_count <= 4) and (dst_bytes >= 1) and (hot <= 0) and (dst_host_rerror_rate <= 0.01) and (num_root <= 0) and (logged_in >= 1) => class=normal (34.0/2.0)*
26) *(dst_host_rerror_rate <= 0) and (dst_host_serror_rate <= 0.97) and (src_bytes <= 706) and (protocol_type = tcp) and (dst_host_same_src_port_rate <= 0.07) and (dst_host_srv_serror_rate >= 0.33) and (count <= 1) => class=normal (45.0/0.0)*
27) *(count <= 52) and (src_bytes >= 30) and (src_bytes <= 92) and (protocol_type = icmp) and (dst_host_diff_srv_rate <= 0.02) => class=normal (19.0/0.0)*
28) *(count <= 5) and (service = finger) and (dst_bytes >= 135) and (srv_count <= 2) => class=normal (91.0/0.0)*
29) *(count <= 7) and (dst_host_srv_diff_host_rate <= 0.1) and (dst_host_count <= 44) and (service = other) and (flag = REJ) => class=normal (8.0/0.0)*
30) *(count <= 7) and (src_bytes >= 9) and (num_root >= 1) and (dst_host_srv_count >= 11) => class=normal (21.0/0.0)*
31) *(count <= 7) and (src_bytes >= 9) and (dst_host_srv_count <= 4) and (src_bytes <= 17) and (dst_host_diff_srv_rate <= 0.27) and (hot <= 0) => class=normal (18.0/0.0)*
32) *(count <= 7) and (src_bytes >= 19) and (dst_bytes <= 2077) and (dst_host_srv_diff_host_rate <= 0.09) and (logged_in >= 1) and (dst_host_same_src_port_rate <= 0.09) and (duration <= 68) and (hot >= 2) => class=normal (13.0/0.0)*
33) *(count <= 7) and (dst_host_srv_diff_host_rate <= 0.1) and (dst_host_count <= 45) and (hot <= 0) and (src_bytes <= 203) and (src_bytes >= 102) => class=normal (28.0/0.0)*
34) *(count <= 11) and (dst_host_same_src_port_rate <= 0.56) and (duration >= 2) and (dst_bytes >= 13020) => class=normal (11.0/0.0)*
35) *(count <= 7) and (dst_host_same_src_port_rate <= 0.56) and (src_bytes >= 5) and (dst_bytes >= 252) and (dst_bytes <= 482) and (dst_host_diff_srv_rate <= 0.07) and (duration <= 7) => class=normal (7.0/0.0)*
36) *(count <= 11) and (service = ftp_data) and (dst_host_srv_diff_host_rate <= 0.09) and (src_bytes >= 383) and (duration <= 2) => class=normal (23.0/0.0)*
37) *(count <= 10) and (dst_host_same_src_port_rate <= 0.55) and (dst_host_rerror_rate <= 0.04) and (serror_rate <= 0.5) and (hot <= 0) and (protocol_type = tcp) and (dst_host_srv_count >= 2) and (src_bytes <= 0) and (dst_host_srv_serror_rate <= 0.26) => class=normal (81.0/0.0)*
38) *(diff_srv_rate <= 0) and (protocol_type = tcp) and (count >= 29) and (service = http) => class=normal (27.0/0.0)*

39) *(count <= 4) and (dst_host_same_src_port_rate <= 0.54) and (dst_host_rerror_rate <= 0) and (dst_host_srv_count <= 1) and (src_bytes >= 30) and (dst_host_diff_srv_rate >= 0.09) and (dst_host_diff_srv_rate <= 0.67) and (dst_host_count <= 225) => class=normal (16.0/0.0)*
40) *(count <= 58) and (src_bytes >= 10) and (service = domain_u) => class=normal (27.0/0.0)*
41) *(count <= 4) and (protocol_type = tcp) and (dst_host_count <= 44) and (src_bytes <= 304) and (dst_host_diff_srv_rate <= 0.43) and (dst_host_same_srv_rate <= 0.8) and (dst_host_serror_rate <= 0.06) => class=normal (18.0/0.0)*
42) *(count <= 58) and (src_bytes >= 38) and (src_bytes <= 319) and (dst_host_count >= 251) and (duration <= 0) and (dst_host_diff_srv_rate <= 0.04) and (root_shell <= 0) => class=normal (12.0/0.0)*
43) *(count <= 57) and (srv_count <= 2) and (dst_host_rerror_rate <= 0.25) and (dst_host_srv_rerror_rate >= 0.05) and (dst_host_same_srv_rate >= 0.03) and (src_bytes <= 244) and (dst_host_diff_srv_rate <= 0.07) => class=normal (9.0/0.0)*
44) *(count <= 7) and (service = ftp_data) and (dst_host_srv_diff_host_rate <= 0.08) and (src_bytes >= 7767) and (duration <= 55) => class=normal (9.0/0.0)*
45) *(count <= 3) and (service = finger) and (src_bytes >= 6) and (srv_count <= 1) => class=normal (8.0/0.0)*
46) *(logged_in >= 1) and (srv_count >= 2) and (src_bytes >= 11) and (src_bytes <= 151) => class=normal (5.0/0.0)*
47) *(count <= 8) and (protocol_type = tcp) and (dst_host_count <= 2) and (src_bytes <= 224) and (src_bytes >= 161) => class=normal (5.0/0.0)*
48) *(count <= 3) and (protocol_type = tcp) and (dst_host_count <= 3) and (logged_in <= 0) and (dst_host_rerror_rate <= 0.5) and (dst_host_srv_serror_rate <= 0.09) and (dst_bytes <= 4) => class=normal (11.0/2.0)*
49) *(count <= 6) and (dst_host_same_src_port_rate <= 0.01) and (dst_host_srv_count <= 6) and (flag = RSTO) => class=normal (12.0/0.0)*

Abstract

An Intrusion Detection System (IDS) is considered as a component that analyses system and user operations in computers and networks in search of activities considered undesirable from security perspectives. Applying mobile agent (MA) for intrusion detection is a recent development, aimed at effectively detecting intrusions in a distributed environment. Previous studies has shown that most available MA-based IDS are not very effective due to their high detection time and limited intrusion detections.

In this thesis, semantic web methods and techniques have been used for distributed intrusion detection. To extract semantic relations between computer attacks and intrusions in that system, a data mined ontology have been utilized. The proposed system contains some Static Agents (Host and Network Sensors), Mobile Agents (Tracer) and a special Central Agent. Central Agent contains the proposed attack ontology. Every time a Static Agent detects an attack or new suspected condition based on its Data Model, it sends detection's report for Central Agent. Therefore, it can extract the semantic relationship among computer attacks and suspected situations in the network with proposed ontology. Jena framework provides the needed interaction between Central Agent (CA) and attacks ontology. The experimental results show that the proposed system reduced the rate of false positive and false negative compared to the similar systems.

Keywords: *Data Mining; Ontology; Intrusion Detection System (IDS); Agent Based IDS*

Department of Computer Engineering

Ferdowsi University of Mashhad

Master's Thesis

Ontology Extraction Based on Data Mining Approach for Cooperative Intrusion Detection

By:

Mehdi Hashemi Shahraki

Supervisor:

Dr. Mohsen Kahani

Advisor:

Prof. Mahmood Naghibzadeh

January 2010