گاساتا(GASATA) یک الگوریتم ژنتیکی به عنوان ابزاری جایگزین برای تحلیل آزمون های حسابرسی امنیت

چکیده

کارایی حسابرسی امنیت پایین است زیرا مامور امنیت بایستی حجم عظیمی از داده های ثبت شده در آزمون حسابرسی را مدیریت کند به طوری که انجام این کار توسط انسان امری کاملا غیر ممکن است.

بنابر این، هدف ما طراحی یک ابزار خودکار برای افزایش کارایی تحلیل آزمون حسابرسی امنیت می باشد. این ابزار که موسوم به GASATA (گاساتا) میباشد(اختصار عبارت الگوریتم زنتیکی برای تحلیل آزمون های حسابرسی امنیتی ساده شده)، بایستی به عنوان یک ابزار اضافی و کمکی در مجموعه ای از ابزار ها در نظر گرفته شود که به مامور امنیتی امکان کنترل دقیق نفوذی های بالقوه را می دهد.

ایده های اصلی که این مطالعه بر اساس آن ها انجام شده است به شرح زیر است:1- تشخیص ناهنجاری(یعنی پاسخ به سوال این که، آیا رفتار کاربر بر طبق گذشته نرمال است یا خیر؟) توسط ابزار هایی نظیر NIDES در نظر گرفته می شود از این روی ما به بررسی تشخیص سو استفاده(یعنی پاسخ به این سوال که آیا رفتار کاربران با یک حمله مشخص موسوم به سناریوی حمله، متناظر است) میپردازیم 2- ما بایستی نفوذ ها به شبکه ناهمگن که بر روی آن ایجاد زمان جهانی غیر ممکن است را تشخیص دهیم به این ترتیب ما بعد زمانی سناریو های حمله را حذف می کنیم(یعنی دلیل این که چرا ما تحلیل خود را با صفت" ساده" توصیف می کنیم)، 3- رویکرد ما از این حیث بد بینانه است که ما تلاش می کنیم تا داده های موجود در آزمون حسابرسی را با یک یا چند حمله توضیح دهیم 4- این مسئله، NP- کامل است از این روی ما از روش اکتشافی، الگوریتم ژنتیکی برای حل آن استفاده می کنیم.

مقاله ما به صورت زیر سازمان دهی شده است. بخش 1 مروری بر مسئله تحلیل آزمون حسابرسی امنیت دارد. در بخش 2، ما نشان می دهیم که چگونه از الگوریتم های زنتیکی در این مسئله استفاده می شود. بخش 3 به بحث در مورد آزمایشاتی می پردازد گه نتایج نسبتا خوبی را نشان می دهند. در نهایت، بخش 4 شامل نتیجه گیری و پیشنهادات برای کار های آینده است.

1-دیدگاه ما در خصوص مسئله تحلیل آزمون حسابرسی امنیت

به طور رسمی، رویکرد را می توان با گزاره زیر بیان کرد:

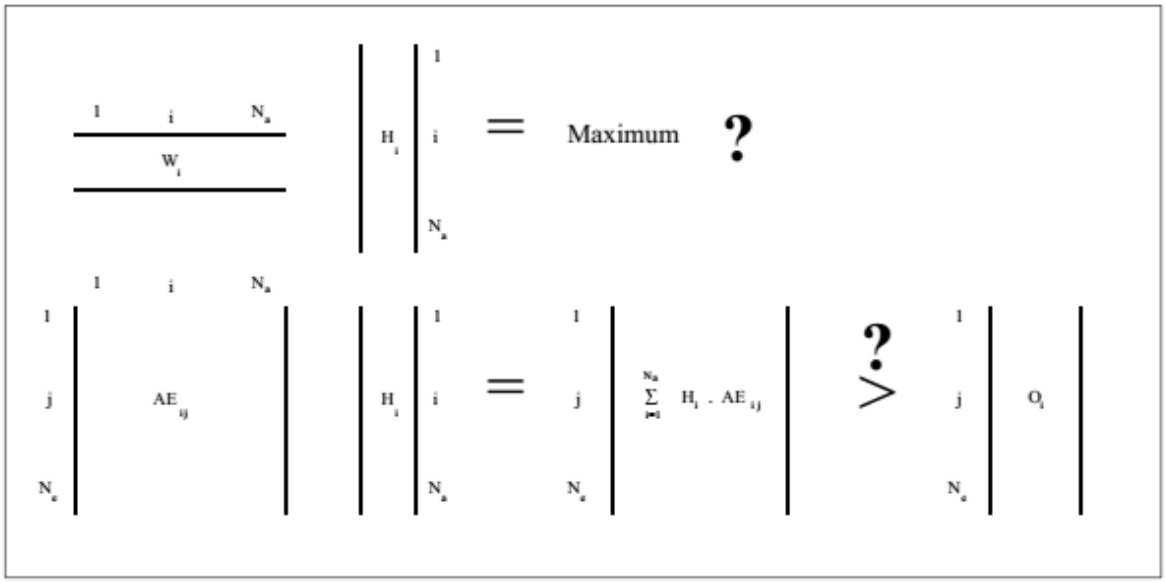
- فرض کنید N_e تعداد رویداد های حسابرسی و N_a تعداد حملات معلوم است.
- فرض کنید که AE، ماتریس حملات-رویداد $N_e \times N_a$ باشد که مجموعه ای از رویداد های تولید شده توسط هر حمله را بدست می دهد. AE_{ij} تعداد رویداد های حسابرسی نوع i می باشد که توسط سناریوی $j\ (AE_{ij} \geq 0)$ تولید شده است(به شکل 4، برای مشاهده این ماتریس مراجعه کنید)
- فرض کنید R، بردار وزنی Na-بعدی باشد که در آن، $R_i\ (R_i > 0)$ وزن مربوط به حمله i است(R_i متناسب با ریسک سناریوی حمله i است)
- فرض کنید که O بردار N_e- بعدی باشد که در آن O_i تعداد رویداد های نوع i موجود در ازمون حسابرسی است(o موسوم به بردار حسابرسی مشاهده شده است).
- فرض کنید که H بردار فرضیه Na- بعدی باشد که در آن $H_i = 1$ است اگر حمله i بر طبق فرضیه موجود باشد ودر غیر این صورت $H_i = 0$ باشد(H بیانگر زیر مجموعه حمله خاص است).

به منظور توضیح داده های موجود در آزمون حسابرسی(O) با وقوع یک یا چند حمله، بایستی بردار H را بیابیم که حاصلضرب $R \times H$ را بیشینه می کند(رویکرد بدبینانه: یافتن H به طوری که ریسک، منوط به $(AE.H)_i \leq O_i,\ (1 \leq i \leq N_a)$ بیشترین باشد(به شکل 1 مراجعه کنید).

بردار H قابل کاهش به مسئله برنامه نویسی عدد صحیح صفر و یک می باشد که موسوم به NP- کامل است. کاربرد الگوریتم های کلاسیک غیر ممکن است که در آن N_a برابر با چند صد است.

رویکرد اکتشافی انتخاب شده برای حل مسئله NP-کامل، به شرح زیر است. یک فرضیه ایجاد شده(برای مثال در میان مجموعه ای از حملات احتمالی، حملات i-j-k در آزمون وجود دارند)، واقعی بودن فرضیه ارزیابی شده و بر طبق این ارزیابی، یک فرضیه اصلاح شده ازمایش می شود تا زمانی که یک راه حل پیدا شود.

به منظور ارزیابی یک فرضیه متناظر با زیر مجموعه ای از حملات موجود، تعداد رویداد های مربوط به هر نوع تولید شده توسط همه حملات فرضیه را شمارش می کنیم. در صورتی که این تعداد کوچکتر مساوی با تعداد روی های ثبت شده در ازمون باشد، آنگاه فرضیه واقعی خواهد بود. مسئله آخر ، یافتن الگوریتمی برای ایجاد فرضیه جدید بر اساس فرضیه گذشته است. این نقش الگوریتم ژنتیکی است.



شکل 1: دیدگاه ما در خصوص مسئله تحلیل ازمون حسابرسی امنیت

2- استفاده از الگوریتم های ژنتیکی برای تشخیص سوء استفاده

الگوریتم های ژنتیکی(GA)، الگوریتم های جست و جوی بهینه بر اساس مکانیسم انتخاب طبیعی در یک جمعیت می باشد. یک جمعیت، مجموعه ای از موجودات مصنوعی(افراد یا کروموزوم ها) می باشد. این موجودات، رشته های به طول 1 می باشند که یک راه حل بالقوه را برای حل مسئله کد گذاری می کنند که اغلب اوقات کد گذاری با یک الفبای دو دویی صورت می گیرد. اندازه L جمعیت، ثابت است. جمعیت به طور تصادفی ایجاد شده و سپس تکامل می یابد: در هر نسل، یک مجموعه جدیدی از موجودات مصنوعی، با استفاده از تواناترین یا بخش هایی از توانمند ترین افراد، ایجاد می شود. توانایی هر فرد معمولا تابع بهینه سازی شده(تابع برازش) برای نقطه متناظر با فرد است. فرایند تکراری خلق جمعیت با سه اپراتور(عملگر) زنتیکی پایه بدست می اید: انتخاب(انتخاب مناسب ترین افراد)، تولید مثل یا کراس اور(بهبود کشف مناطق جدید فضای

جست و جو با کراسینگ اور بخش هایی از افراد) و موتاسیون(حفاظت از جمعیت در برابر از دست رفت اطلاعات). ساختار عمومی GA به شرح زیر است:

تولید تصادفی اولین نسل

تکرار

انتخاب فرد

تولید مثل

موتاسیون

تا زمانی که معیار های توقف بدست بیایند.

اپراتور های(عملگر های) ژنتیکی، عملگر های تصادفی می باشند، با این حال الگوریتم های ژنتیکی پیمایش های تصادفی ساده نمی باشند: آن ها به طور کارامد از اطلاعات تاریخی برای فرض نقاط جست و جوی جدید با عملکرد مورد انتظار استفاده می کنند.

دو زیر مسئله در زمان کاربرد GA به یک مسئله ویژه حادث می شوند: 1- کد گذاری یک راه حل برای آن مسئله با یک رشته بیتی 2- یافتن یک تابع برازش برای ارزیابی هر فرد از جمعیت.

2-1 کد گذاری یک راه حل با رشته دو دویی

یک فرد، یک رشته به طول 1 است که یک راه حل بالقوه را برای مسئله کد گذاری می کند. در این مورد، کد گذاری ساده است: طول یک فرد Na می باشد و هر فرد در جمعیت متناظر با بردار h می باشد که در بخش 1 تعریف شده است.

2-2 تابع برازش

ما بایستی در میان همه زیر مجموعه حملات احتمالی به جست وجوی حمله ای بپردازیم که بیشترین خطر را برای سیستم دارد. این مسئله منجربه بیشینه سازی حاصلضرب R.H می شود. از آن جا که GA، الگوهای جست و جوی بهینه ای میباشند، با یافتن حداکثر تابع برازش می توان به آسانی نتیجه گرفت که در این مورد، تابع بایستی برابر با حاصلضرب R.H باشد. از این روی داریم

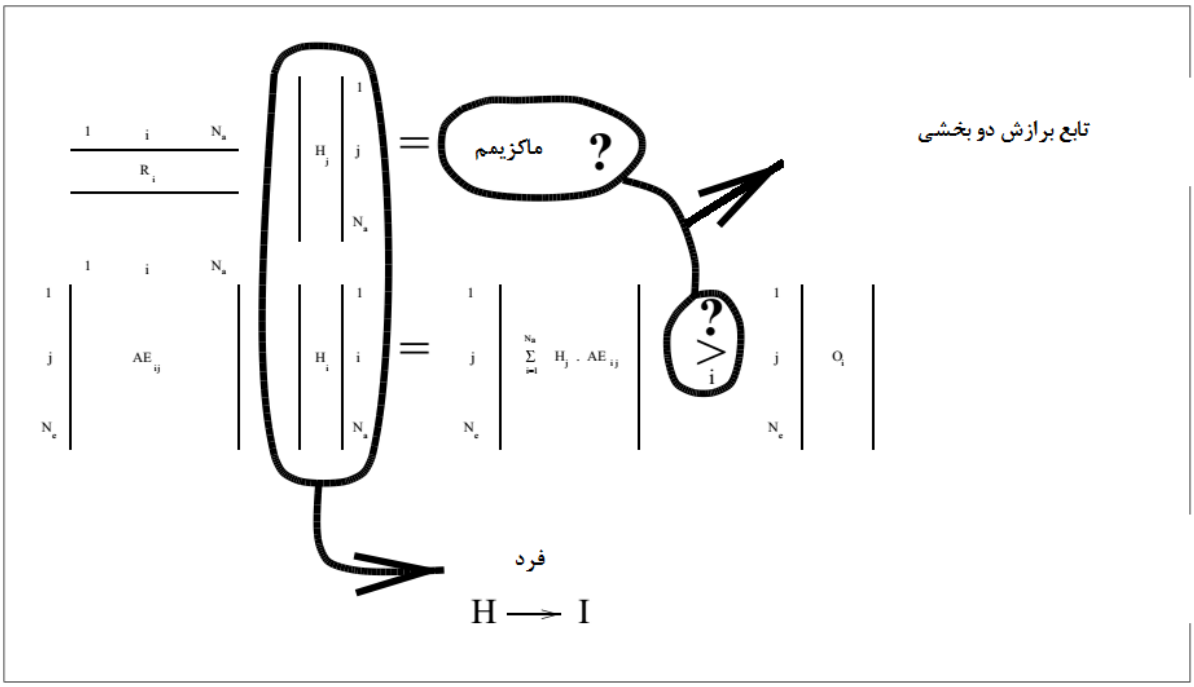
$$\sum_{i=1}^{Na} R_i.I_i$$

برازش:

که I فرد است.

با این حال تابع برازش، ویژگی محدود کننده(یا قیدی) مسئله ما را در نظر نمیگیرد و این به طور ضمنی نشان می دهد که برخی فرضیات(یعنی برخی از افراد) در میان افراد $\bar{2}^{Na}$، واقع گرایانه نمی باشند. این مسئله در رویداد های نوع i صادق است به خصوص زمانی که $(\dot{AE}.H)_i > O_i$ باشد. از آن جا که تعداد زیادی از افراد محدودیت را در نظر نمی گیرند، ما تصمیم گرفتیم تا آن ها را با کاهش ارزش برازششان جریمه کنیم. از این روی ما تابع جرمیه را محاسبه می کنیم که با کاهش واقع گرایی این فرد، افزایش می یابد.فرض کنید که T_ϵ تعداد رویداد هایی باشد که به ازای آن ها $(AE.H)_i > O_i$; وجود دارد و تابع جریمه ای قابل کاربرد به این فرد H است سپس

$$P = T_\epsilon{}^p$$



شکل 2: یک تابع برازش برای استفاده از الگوریتم های ژنتیکی برای تشخیص سو استفاده

یک تابع جریمه ای دو جمله ای(P=2)، امکان تمایز خوب را در میان افراد می دهد. تابع برازش پیشنهادی به شرح زیر است

$$F(I_i) = \alpha + \left(\sum_{i=1}^{N_a} R_i.I_i \; - \beta.T_\epsilon{}^2 \right)$$

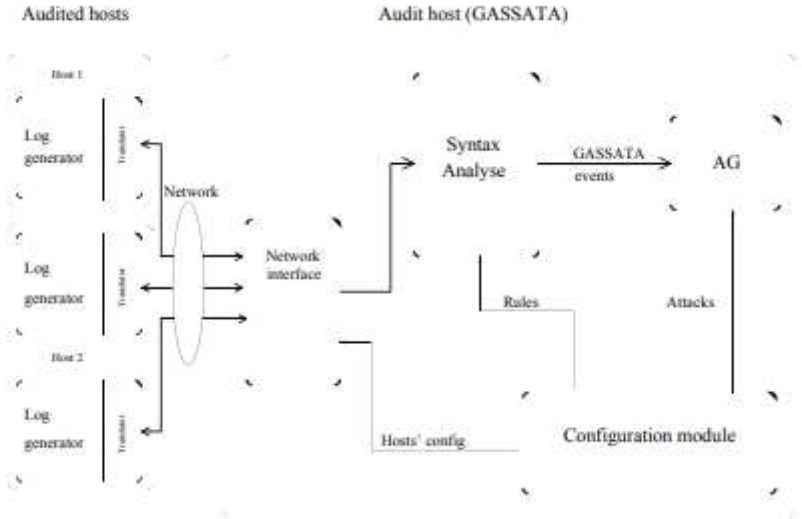
پارامتر β امکان اصلاح شیب تابع جریمه ای را داده و مقدار α یک استانه ای را تنظیم می کند که موجب می شود تا برازش مثبت باشد. در صورتی که یک مقدار برازش منفی یافت شود، این برابر با صفر است و فرد متناظر را نمی توان انتخاب کرد. از این روی پارامتر α امکان حذف دو فرضیه غیر واقعی را می دهد.

به شکل 2-2 مراجعه کنید که گزینه های افراد و تابع برازش را نشان می دهد.

3-نتایج ازمایشی برای حملات و کاربران شبیه سازی شده

3-1 محیط ازمایشی

سیستم AIX، یک زیر سیستم حسابرسی امنیتی را ارایه می کند که امکان تولید انواع مختلف رویداد های حسابرسی را می دهد که در یک فایل حفاظت شده ثبت می شود. برای این ازمایشات، ما از این زیر سیستم حسابرسی استفاده می کنیم.



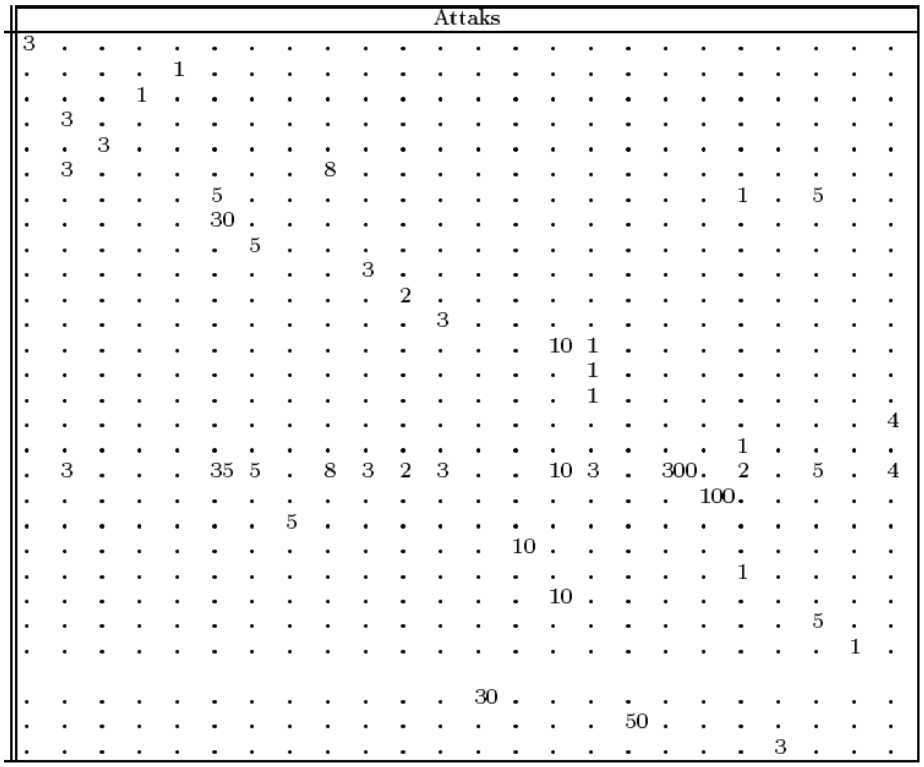
شکل 3: معماری گاساتا

نمونه ما، یعنی گاساتا GASAT(به شکل 3 نگاه کنید که معماری آن را نشان می دهد) یک مقدار بردار H را پیدا می کند که موجب بیشینه سازی حاصلضرب R.H منوط به $(AE.H)_i \leq O_i \; (1 \leq i \leq N_a)$

می شود. در صورتی که جلسه حسابرسی بسیار طولانی باشد، این محدودیت همیشه اجرا شده و گاساتا بر روی بردار واحد Na- بعدی همگرا می شود. برای اجتناب از این مسئله، مدت زمان جلسه حسابرسی بایستی به طور دقیق انتخاب شود. به همین دلیل ما بر اساس جلسات حسابرسی 30 دقیقه ای کار می کنیم. ما ازمون حسابرسی را به بردار های حسابرسی کاربر به کاربر با الگوریتم خطی تک مسیره ترجمه می کنیم(در یک محیط واقعی، ازمون های حسابرسی متوالی و بردار های حسابرسی بایستی برای تحقیقات آینده ضبط شوند).

به منظور انجام آزمایشات واقعی بر روی انواع کاربران، ما چهار نوع کاربر زیر را تعریف کرده ایم: کاربر بی تجربه، توسعه دهنده مبتدی، توسعه دهنده حرفه ای و کاربر UNIX. هر نوع کاربر، با یک مجموعه و توالی ای از دستورات تعریف می شودکه توسط این کاربر در طی یک دوره سی دقیقه ای کامل می شود. هر یک از این توالی ها به یک بردار حسابرسی مشاهده شده ترجمه می شوند. مایک ماتریس حملات-رویداد از جمله 24 حمله متفاوت را طراحی کردیم. یک مثال از یک حمله با دستورات زیر بدست می اید که به مهاجم امکان پرینت هر فایل را می دهد(قبل از مرحله 2، مهاجم بایستی اطمینان حاصل کند که شغل ها در صف چاپ قرار دارند).

```
> touch f
> lpr -s f
```



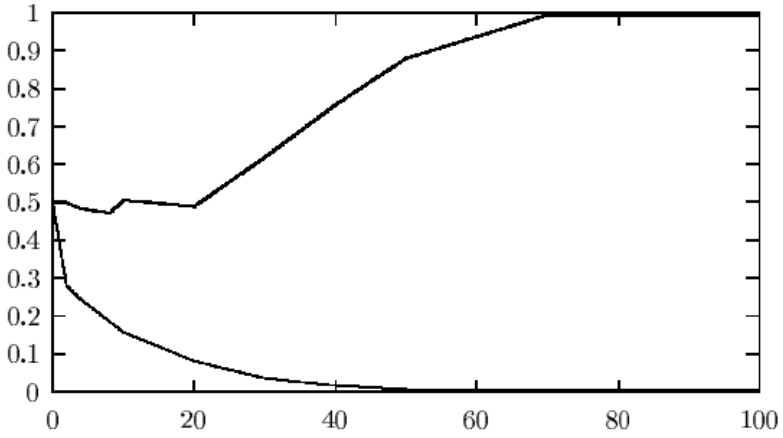
شکل 4: یک مثال از ماتریس حملات

```
> rm f
> ln -s /etc/security/passwd f
```

تعداد رویداد های ناشی از 24 حمله برابر با 28 است(از این روی ما یک ماتریس 24×28 را داریم به شکل 4 مراجعه کنید).(ما از ماتریس 200×28 برای تعیین عملکرد و کارایی گاساتا از حیث زمان های اجرا استفاده کردیم، به بخش 3-2 و شکل 7 و 8 مراجعه کنید).

به منظور درک نتایج ارایه شده توسط گاساتا، ما بایستی از قبل، با مجموعه حملاتی که واقعا در آزمون حسابرسی تحلیل شده وجود دارند آشنا باشیم. به همین دلیل حملات با قرار دادن رویداد های متناظر با یک یا چند حمله در بردار های حسابرسی مشاهده شده شبیه سازی می شوند. به علاوه، ما بایستی نتایج گاساتا را زمانی بررسی کنیم که هیچ حمله ای در آزمون گنجانده نشده باشد.

هر آزماش انجام شده دارای چهار تاپل (P_c, P_m, L, a) می باشد که در آن Pc احتمال کراس اور،Pm احتمال موتاسیون، L، اندازه جمعیت و a تعداد حملاتی است که واقعا در ازمون حسابرسی وجود دارند. برای هر مقدار از 4 تاپل، ما 10 تکرار را اجرا می کنیم(همه نتایج زیرمیانگین ده تکرار هستند). از دیدگاه ژنتیکی، تحلیل اثر هر پارامتر جالب می باشد. از دیدگاه امنیتی، کیفیت تشخیص نیز مهم تر است. ما در این جا بر دیدگاه امنیتی تاکید کرده و برای درک کیفیت تشخیص، دو نسبت T_p وT_a را به صورت زیر تعریف می کنیم.



شکل 5: تغییرات T_p و T_a در برابر نسل

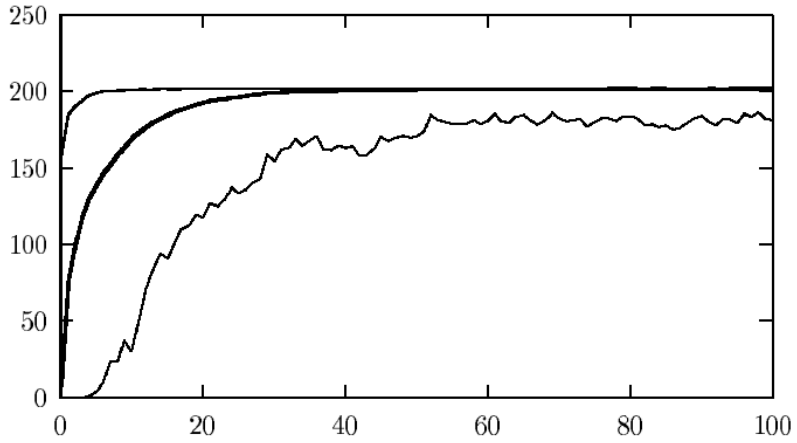
- Tp تعداد افرادی است که در آن ها بیت های متناظر با حملات موجود، 1 از کل تعداد افراد است
- Ta تعداد افرادی است که در آن ها بیت های متناظر با حملات غایب، 1 از کل تعداد افراد است

توجه داشته باشید که T_{p_i} و T_{a_i}، مقادیر T_p و T_a برای نسل i می باشد. از این روی داریم: $T_{p_0} \simeq 0.5$ و $T_{a_0} \simeq 0.5$ (جمعیت اولیه به صورت تصادفی تولید می شود) و بایستی دارای $T_{p_{final}} = 1$ و $T_{a_{final}} = 0$ باشد(همه حملات موجود تشخیص داده شده و هیچ جمله غایبی تشخیص داده نمی شود).

به منظور درک همگرایی جمعیت و زمان مورد نیاز برای یک تشخیص، ما برای هر نسل، مقدار حداقل، حداکثر و میانگین مقادیر برازش جمعیت را حساب می کنیم.

2-3 نتایج

شکل 5 تغییرات T_p و T_a را در برابر تعداد نسل(یعنی در برابر زمان) نشان می دهد. این نشان می دهد که یک تمایز خوب بین حملات حاضر و غایب وجود دارد: مقادیر میانگین $T_{p_{100}}$ و $T_{a_{100}}$ به ترتیب 0.996 و 0.0044 است. از این روی ما نزدیک به مقادیر بهینه 1 و 0 هستیم. تعداد حملات موجود در ازمون اثری بر روی این نتیجه ندارد.

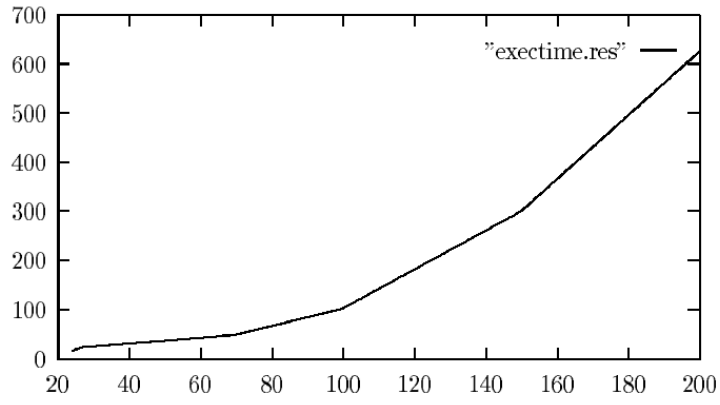


شکل 6: حداقل و حداکثر برازش متوسط برای 10 تکرار($\lambda = 500,\ P_c = 0.6,\ P_m = 0.002$ و $a = 2$) در برابر نسل

شکل 6 نشان می دهد که ماکزیمم مقدار برازش سریعا پس از 20 نسل بر روی ماتریس حملات- رویداد ها همگرا می شوند. باقی مانده جمعیت پس از 100 نسل به صورت زیر است: برازش میانگین حدود 99 درصد برازش ماکزیمم است. مجددا لازم به ذکر است که تعداد حملات موجود در ازمون اثری بر روی این نتیجه ندارند.

در صورتی که تعداد حملات کد گذاری شده در ماتریس حملات- رویداد ها رشد کند، تعداد نسل نهایی بایستی افزایش پیدا کند تا کیفیت تشخیص در سطح یکسانی باقی بماند(یعنی $T_{p_{final}}$ و $T_{a_{final}}$ نزدیک به مقادیر بهینه هستند). شکل 7 تغییرات زمان اجرا را با رشد تعداد حملات نشان می دهد با در نظر گرفتن 200 حمله، گاساتا نیازمند 10 دقیقه و 25 ثانیه برای ارایه نتایج تحلیل است. شکل 8 تغییرات تعداد راه حل های احتمالی و تغییرات زمان اجرا را بر حسب ثانیه نشان می دهد به خصوص زمانی که تعداد حملات کد گذاری شده در ماتریس رشد کند. اولین مورد به طور نمایی رشد می کند و دومین مورد به طور چند جمله ای رشد می کند. با در نظر گرفتن 24 حمله، تعداد فرضیات ازمایش شده از تعداد کل راه حل احتمالی ، 0.003 است. با 100 حمله، این عدد برابر با 5.9×10^{-26} و با 200 حمله برابر با 7.7×10^{-54} است. این نشان می دهد که الگوریتم های زنتیکی، یک روش اکتشافی قوی را ایجاد می کنند.

در نهایت، لازم به ذکر است که مدت زمان جلسه حسابرسی بایستی بر زمان اجرا اثر داشته باشد زیرا تنها بستگی به اندازه دو ماتریس ثابت H و AE دارد.



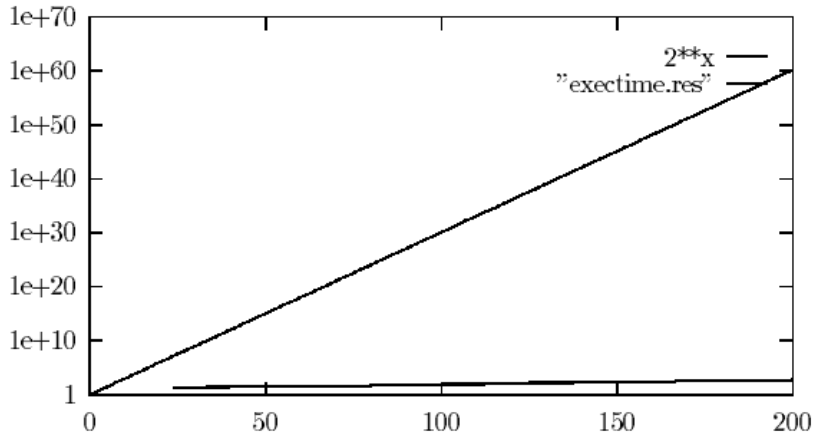
شکل 7: زمان اجرا بر حسب ثانیه در برابر تعداد حملا در ماتریس

4-کار های آینده

برخی از مسائل باقی مانده(ناشی از استفاده از سناریو های حمله از پیش تعریف شده و یا دیدگاه ساده ما نسبت به مسئله)، می توانند انگیزه ای برای انجام مطالعات اینده باشد

- ما قادر به در نظر گرفتن حملات با یک رویداد غایب نیستیم(برای مثال، یک برنامه نویس که از کمپایلر CC استفاده می کند)
- با استفاده از کد گذاری دو دویی برای افراد، ما قادر به تشخیص تحقق چندگانه یک حمله خاص نمی باشیم. در نتیجه، ما بایستی GA غیر دو دویی را ازمایش کنیم.
- در صورتی که یک رویداد یا گروهی از رویداد ها در چندین حمله رخ دهند، یک نفوذی تشخیص دهنده این حملات به طور همزمان قادر به تکرار این رویداد یا گروه از رویداد ها نیست. در این صورت، گاساتا قادر به یافتن بردار H بهینه نیست. ما راه حلی برای این مسئله نداریم. این بدین معنی است که ما تنها رویداد های مستقل را در نظر می گیریم.
- گاساتا قادر به مکان یابی دقیق حملات در ازمون حسابرسی نیست. همانند ابزار های تشخیص نفوذی اماری، تنها یک مجموعه ای از حملات موجود در یک جلسه حسابرسی معین را بدست می دهد. ازمون حسابرسی بایستی توسط یک افسر امنیتی برای مکان یابی دقیق حملات بررسی شود.

ازمایشات ما، شبیه سازی های نسبتا ساده ای هستند. این اولین مرحله از ازمایشات می باشد و هدف ما استفاده از گاساتا در یک محیط واقعی است.



شکل 8: تعداد راه حل های احتمالی و زمان اجرا بر حسب ثانیه در برابر تعداد حملات در ماتریس

برای یک جامعه متشکل از 100 تا 200 کاربر واقعی که تولید داده های حسابرسی واقعی می کنند ما به دنبال حملات بالقوه واقعی هستیم. دومین مرحله از ازمایش به ما امکان مقایسه عملی ابزار ما را با ابزار های دیگر می دهد.

برای کسب اطلاعات بیشتر در این زمینه http://www.supelec-rennes.fr/rennes/si/equipe/lme/these/these-lm.html را ببینید.